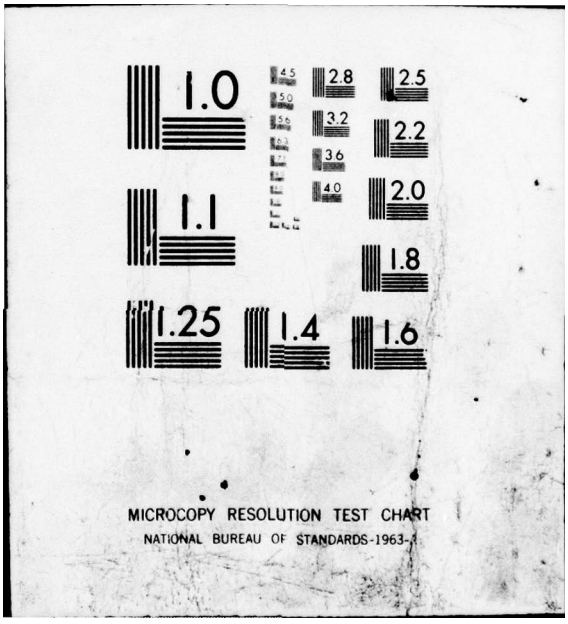
AD-A076 875 STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE F/G 9/4
APPLICATIONS-ORIENTED AI RESEARCH: SCIENCE AND MATHEMATICS.(U)
AUG 79 J S BENNETT , B G BUCHANAN MDA903-77-C-0322
UNCLASSIFIED STAN-CS-79-756 NL

1 OF 2
AD
A076875


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Stanford Heuristic Programming Project
Memo HPP-79-22

Computer Science Department
Report No. STAN-CS-79-756

August 1979

LEVEL II

Applications-oriented AI Research: Science and Mathematics

by

James S. Bennett, Bruce G. Buchanan, Paul R. Cohen, and Fritz Fisher

a section of the

Handbook of Artificial Intelligence

edited by

Avron Barr and Edward A. Feigenbaum

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: HPP-79-22 (STAN-CS-79-756, HPP-79-22)	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Applications-oriented AI Research: Science and Mathematics (a section of the Handbook of Artificial Intelligence)	5. TYPE OF REPORT & PERIOD COVERED: technical, rept.
	6. PERFORMING ORG. REPORT NUMBER: HPP-79-22 (STAN-CS-79-756)
7. AUTHOR(s): James S. Bennett, Bruce G. Buchanan, Paul R. Cohen, and Fritz Fisher (A. Barr and E. A. Feigenbaum, editors)	8. CONTRACT OR GRANT NUMBER(s): ARPA MDA 903-77-C-0322; NIH RR-00785-06 PHS
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Computer Science, Stanford University, Stanford, California 94305 USA	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 12 1991
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, Information Processing Techniques Office, 1400 Wilson Ave., Arlington, VA 22209	12. REPORT DATE: August 1979
	13. NUMBER OF PAGES: 106
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Mr. Philip Surra, Resident Representative, Office of Naval Research, Durand 165, Stanford University	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Reproduction in whole or in part is permitted for any purpose of the U.S. Government.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(see reverse side)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 094 120

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Those of us involved in the creation of the Handbook of Artificial Intelligence, both writers and editors, have attempted to make the concepts, methods, tools, and main results of artificial intelligence research accessible to a broad scientific and engineering audience. Currently, AI work is familiar mainly to its practicing specialists and other interested computer scientists. Yet the field is of growing interdisciplinary interest and practical importance. With this book we are trying to build bridges that are easily crossed by engineers, scientists in other fields, and our own computer science colleagues.

In the Handbook we intend to cover the breadth and depth of AI, presenting general overviews of the scientific issues, as well as detailed discussions of particular techniques and important AI systems. Throughout we have tried to keep in mind the reader who is not a specialist in AI.

As the cost of computation continues to fall, new areas of computer applications become potentially viable. For many of these areas, there do not exist mathematical "cores" to structure calculational use of the computer. Such areas will inevitably be served by symbolic models and symbolic inference techniques. Yet those who understand symbolic computation have been speaking largely to themselves for twenty years. We feel that it is urgent for AI to "go public" in the manner intended by the Handbook.

Several other writers have recognized a need for more widespread knowledge of AI and have attempted to help fill the vacuum. Lay reviews, in particular Margaret Boden's Artificial Intelligence and Natural Man, have tried to explain what is important and interesting about AI, and how research in AI progresses through our programs. In addition, there are a few textbooks that attempt to present a more detailed view of selected areas of AI, for the serious student of computer science. But no textbook can hope to describe all of the sub-areas, to present brief explanations of the important ideas and techniques, and to review the forty or fifty most important AI systems.

The Handbook contains several different types of articles. Key AI ideas and techniques are described in core articles (e.g., basic concepts in heuristic search, semantic nets). Important individual AI programs (e.g., SHRDLU) are described in separate articles that indicate, among other things, the designer's goal, the techniques employed, and the reasons why the program is important. Overview articles discuss the problems and approaches in each major area. The overview articles should be particularly useful to those who seek a summary of the underlying issues that motivate AI research.

Eventually the Handbook will contain approximately two hundred articles. We hope that the appearance of this material will stimulate interaction and cooperation with other AI research sites. We look forward to being advised of errors of omission and commission. For a field as fast moving as AI, it is important that its practitioners alert us to important developments, so that future editions will reflect this new material. We intend that the Handbook of Artificial Intelligence be a living and changing reference work.

The articles in this edition of the Handbook were written primarily by graduate students in AI at Stanford University, with assistance from graduate students and AI professionals at other institutions. We wish particularly to acknowledge the help from those at Rutgers University, SRI International, Xerox Palo Alto Research Center, MIT, and the RAND Corporation.

The authors of this report, which contains the section of the Handbook describing research on applying AI techniques to systems in science and mathematics, are James Bennett, Bruce Buchanan, Paul Cohen, and Fritz Fisher. Others who contributed to or commented on earlier versions of this section include Randall Davis, Daniel Dolata, Richard Duda, Robert Engelmore, Peter Friedland, Michael Genesereth, Douglas Lenat, and Glen Ouchi.

Applications-oriented AI Research: Science and Mathematics

by

James S. Bennett, Bruce G. Buchanan, Paul R. Cohen, and Fritz Fisher

a section of the

Handbook of Artificial Intelligence

edited by

Avron Barr and Edward A. Feigenbaum

This research was supported by both the Defense Advanced Research Projects Agency (ARPA Order No. 3423, Contract No. MDA 903-77-C-0322) and the National Institutes of Health (Contract No. NIH RR-00785-06). The views and conclusions of this document should not be interpreted as necessarily representing the official policies, either express or implied, of the Defense Advanced Research Projects Agency, the National Institutes of Health, or the United States Government.

Applications-oriented AI Research: Science and Mathematics

Table of Contents

Foreword

Those of us involved in the creation of the Handbook of Artificial Intelligence, both writers and editors, have attempted to make the concepts, methods, tools, and main results of artificial intelligence research accessible to a broad scientific and engineering audience. Currently, AI work is familiar mainly to its practicing specialists and other interested computer scientists. Yet the field is of growing interdisciplinary interest and practical importance. With this book we are trying to build bridges that are easily crossed by engineers, scientists in other fields, and our own computer science colleagues.

In the Handbook we intend to cover the breadth and depth of AI, presenting general overviews of the scientific issues, as well as detailed discussions of particular techniques and important AI systems. Throughout we have tried to keep in mind the reader who is not a specialist in AI.

As the cost of computation continues to fall, new areas of computer applications become potentially viable. For many of these areas, there do not exist mathematical "cores" to structure calculational use of the computer. Such areas will inevitably be served by symbolic models and symbolic inference techniques. Yet those who understand symbolic computation have been speaking largely to themselves for twenty years. We feel that it is urgent for AI to "go public" in the manner intended by the Handbook.

Several other writers have recognized a need for more widespread knowledge of AI and have attempted to help fill the vacuum. Lay reviews, in particular Margaret Boden's **Artificial Intelligence and Natural Man**, have tried to explain what is important and interesting about AI, and how research in AI progresses through our programs. In addition, there are a few textbooks that attempt to present a more detailed view of selected areas of AI, for the serious student of computer science. But no textbook can hope to describe all of the sub-areas, to present brief explanations of the important ideas and techniques, and to review the forty or fifty most important AI systems.

The Handbook contains several different types of articles. Key AI ideas and techniques are described in core articles (e.g., basic concepts in heuristic search, semantic nets). Important individual AI programs (e.g., SHRDLU) are described in separate articles that indicate, among other things, the designer's goal, the techniques employed, and the reasons why the program is important. Overview articles discuss the problems and approaches in each major area. The overview articles should be particularly useful to those who seek a summary of the underlying issues that motivate AI research.

Eventually the Handbook will contain approximately two hundred articles. We hope that the appearance of this material will stimulate interaction and cooperation with other AI research sites. We look forward to being advised of errors of omission and commission. For a field as fast moving as AI, it is important that its practitioners alert us to important developments, so that future editions will reflect this new material. We intend that the **Handbook of Artificial Intelligence** be a living and changing reference work.

The articles in this edition of the Handbook were written primarily by graduate students in AI at Stanford University, with assistance from graduate students and AI professionals at other institutions. We wish particularly to acknowledge the help from those at Rutgers University, SRI International, Xerox Palo Alto Research Center, MIT, and the RAND Corporation.

The authors of this report, which contains the section of the Handbook describing research on applying AI techniques to systems in science and mathematics, are James Bennett, Bruce Buchanan, Paul Cohen, and Fritz Fisher. Others who contributed to or commented on earlier versions of this section include Randall Davis, Daniel Dolata, Richard Duda, Robert Engelmore, Peter Friedland, Michael Genesereth, Douglas Lenat, and Glen Ouchi.

Avron Barr
Edward Feigenbaum

Stanford University
July, 1979

Handbook of Artificial Intelligence

Topic Outline

Volumes I and II

Introduction

- The Handbook of Artificial Intelligence
- Overview of AI Research
- History of AI
- An Introduction to the AI Literature

Search

- Overview
- Problem Representation
- Search Methods for State Spaces, AND/OR Graphs, and Game Trees
- Six Important Search Programs

Representation of Knowledge

- Issues and Problems in Representation Theory
- Survey of Representation Techniques
- Seven Important Representation Schemes

AI Programming Languages

- Historical Overview of AI Programming Languages
- Comparison of Data Structures and Control Mechanisms in AI Languages
- LISP

Natural Language Understanding

- Overview - History and Issues
- Machine Translation
- Grammars
- Parsing Techniques
- Text Generation Systems
- The Early NL Systems
- Six Important Natural Language Processing Systems

Speech Understanding Systems

- Overview - History and Design Issues
- Seven Major Speech Understanding Projects

Applications-oriented AI Research -- Part 1

Overview
TEIRESIAS - Issues in Expert Systems Design
Research on AI Applications in Mathematics (MACSYMA and AM)
Miscellaneous Applications Research

Applications-oriented AI Research -- Part 2: Medicine

Overview of Medical Applications Research
Six Important Medical Systems

Applications-oriented AI Research -- Part 3: Chemistry

Overview of Applications in Chemistry
Applications in Chemical Analysis
The DENDRAL Programs
CRYSALIS
Applications in Organic Synthesis

Applications-oriented AI Research -- Part 4: Education

Historical Overview of AI Research in Educational Applications
Issues in ICAI Systems Design
Seven Important ICAI Systems

Automatic Programming

Overview
Techniques for Program Specification
Approaches to AP
Eight Important AP Systems

The following sections of the Handbook are still in preparation and will appear in the third volume:

Theorem Proving
Vision
Robotics
Information Processing Psychology
Learning and Inductive Inference
Planning and Related Problem-solving Techniques

A. Overview

Within the past decade, Artificial Intelligence (AI) techniques have been applied to the development of *expert systems*, computer systems intended to assist researchers solve complex problems in their scientific or medical speciality. These systems are most strongly characterized by their use of *domain knowledge*, gleaned from experts, in the problem-solving tasks.

The systems described here were originally designed to be applied in their intended communities, and all but a few are in consistent use. Most of the systems are still being researched and developed. The emphasis in this chapter is on a description of the applications and research of AI techniques on real-world problems.

A layman or general researcher is distinguished from a specialist in a scientific or technical domain by the vast amount of *empirical* knowledge that the expert has amassed during the course of his profession. This task-specific knowledge is, of course, based on any *conceptual* or theoretical knowledge that underlies problem solving in the domain. Any so-called *knowledge-based* system designed to assist users in the domain at this expert level requires both the empirical and the theoretical knowledge. Developing representational vehicles that are able to encode this partly public, partly private knowledge of the domain has occupied the AI researchers during the construction of all these systems.

Using representations of domain-specific knowledge, artificial intelligence research has yielded systems with significant problem-solving abilities, at times better than the abilities of the human experts. In addition to developing adequate representations of this domain-specific knowledge, research has emphasized the development of various reasoning and explanation procedures that manipulate this knowledge. In particular, much emphasis has been placed on the development of methods of *inexact reasoning* since for many of these domains, notably medicine, the experts' appraisal of the problem situation cannot always be certain.

The major domains of expertise that have been developed as applications systems include: the diagnosis and treatment of various dieases (see section Medicine.C1), the design of computer assisitants for both the analytic and synthetic aspects of organic chemistry (see section C1), interactive tutoring systems in education (see section Education.Overview), and assistants for performing advanced mathematics (see article D2). A number of other notable applications have been developed including applications of AI to database information retrieval problems (see article E4) and a geological assistant (see article E2). There are a host of recent applications as well that do not have articles in this chapter, such as SACON, a system for advising structural engineers in the use of a large finite-element analysis program used to model various mechanical structures (Bennett et al., 1978); PUFF, a system for diagnosing a patient with various pulmonary dysfunctions (Feigenbaum, 1977); and HEADMED, a system for diagnosis and treatment of psychiatric patients (Heiser, 1977, 1978).

Typically, these systems are considered intelligent if they meet the following criteria: The system gives correct answers or useful advice, and the concepts and reasoning processes that the system uses to solve the problem resemble those that the user might employ. This last concern has motivated the design of systems capable of explaining their reasoning about a case, capable of maintaining a focused dialogue with a user when pursuing

relevant facts and inferences about the user's case, and capable of using knowledge at the conceptual level of the user when solving and explaining both the problem and the system's solution. Achieving these primarily *human-engineering* concerns has required many advances in artificial intelligence. These abilities and developments are detailed for each system in the following articles.

Evolution of Expert Systems

Work in AI during the 1960s identified and explored general-purpose problem-solving techniques that would be applicable in a large number of problem-solving situations. This research introduced and refined the concept of heuristic search (see Search) as a mechanism of problem solving. These ideas and developments were embodied in such systems as GPS, REF-ARF, QA4, PLANNER, etc. These systems dealt with problems in domains such as chess, robot planning, and blocks-world manipulations, as well as the classic problem-solving situations found in puzzles such as the Tower of Hanoi and The Missonaries and Cannibals.

During the mid-1960s, the first expert systems were developed, including DENDRAL and MACSYMA. In 1965, the Heuristic Programming Project at Stanford University began to apply these search techniques to the design of an intelligent assistant to aid chemists in elucidating the structure of unknown chemical compounds. Motivated by interest in modeling the thought process of research scientists, Edward Feigenbaum and Joshua Lederberg of the DENDRAL project began to emphasize and use large amounts of domain-specific knowledge in the solution of this major *real-world problem*.

These systems were designed to manipulate and explore large, symbolically expressed problems that were known to be difficult for human researchers to solve. These problems were characterized by the fact that as their specifications grew in complexity, so did the number of solution possibilites that had to be examined. The larger the size of the problem specification (e.g., size of the molecule in atoms/bonds or complexity of the expression to be intergrated), the more difficult it was for human researchers to discover solutions or be confident that all valid solutions had been found. This *combinatorial explosion* in the solution search space easily outstripped the abilities of most human researchers. The ability of these applications systems to deal with the larger solution spaces extended the limit on the types of problems capable of solution with the present conceptual tools.

More recently, the motivation for constructing these knowledge-based systems includes a number of other factors. These expert systems promise to have significant economic and social impact. (See especially the articles on Synthesis and PROSPECTOR). For example, the organic synthesis systems are used actively by drug and chemical manufacturing companies to uncover inexpensive methods of synthesizing various compounds. In medicine, these systems have the capability to examine all possible diseases that might be afflicting a patient. In addition, the ability to codifiy the expertise in a domain makes these systems potentially available for tutoring and assessment purposes.

For a system to achieve broad applicability within a speciality and to remain complete and correct in its search for problem solutions, large amounts of domain-specific knowledge have had to be represented and handled. Thus, while heuristic search management is still a major concern in the construction of any expert system, the large amounts of expert

knowledge required to achieve an adequate, efficient solution to these problems have fostered problems in the construction and maintainence of these knowledge bases. The concerns of effective representation and management of the large, domain-specific knowledge bases have shifted attention away from development of programs designed to solve large combinatorial problems, such as those that prompted the DENDRAL programs, to those that require more empirical knowledge for their solution. Current research emphasizes not only the representational adequacy of the existing formalisms but also such issues as the appropriate *grain size* of the knowledge (see article Representation.B) and improved explanation, inference, and acquisition abilities (B).

Dimensions of Applications

Most of the application systems described in this chapter can be viewed as *consultants* that formulate opinions or as models about *cases* that give advice to their users. The tasks these consultants are designed to perform are typically repetitive and sometimes beyond human abilities--problems that require knowledge of facts and relationships known only by specialists. A consultation system interacts with the user during the problem-solving task. The current systems emphasize the cognitive abilities that support this interaction such as the abilities to explain lines of reasoning or to interactively acquire new domain knowledge. This is especially true for the medical and educational systems where much research has gone into the design of well-engineered, responsive, user interfaces.

The AI research conducted for these application systems is different from other mainstream AI research such as that on speech or vision. Applications research does not concentrate on developing models of the various physiological functions that are of interest in these other areas. The cognitive abilities required by the current applications are primarily conceptual in nature and do not depend on sophisticated perceptual capabilities in order to be performed. Research concentrates instead on the requirements for systems to utilize *developed* human expertise. This expertise is typically at a high conceptual level and is easily encodable in the symbolic representational formalisms that have been developed.

Representational adequacy. Applications research has proved a valuable testing ground for the techniques developed in other areas of AI research. In addition to the augmentation of heuristic search methods by domain-specific knowledge, representation formalisms developed for modeling psychological aspects of cognition--such as semantic nets (see article Representation.C2) and production systems (see article Representation.C3)--have been used ubiquitously in the applications described in this chapter. Techniques developed in the course of natural language research (see chapter Natural Language, Natural Language) have been used to achieve the effective man-machine interface required of these interactive consultant systems.

Domain-independence of the systems. As part of the research on the adequacy of these representational formalisms, a number of these systems have attempted to maintain a strict separation between the domain-specific knowledge supplied by the expert and the domain-independent knowledge and capabilities of problem solving that the systems intrinsically possess. The task of determining what abilities and what knowledge constitutes an effective domain-independent system occupies much of the AI research in applications. For example, the EMYCIN system consists of the basic control structure found in the MYCIN system (see article Medicine.C2) with the infectious disease knowledge base removed; this

"empty MYCIN" system retains the capability to interact with the user during a case, to explain its reasoning, and to answer questions about a case in a new domain of expertise. This system has been used successfully to develop the applications in pulmonary dysfunction, structural analysis, and psychiatric diagnosis mentioned in the beginning of this overview. Numerous other systems similar to the EMYCIN system are being developed, such as the IRIS system (see article Medicine.C7); these domain-independent consultation systems are a major product of this applications research.

Explanation and the opacity of knowledge. As mentioned previously, a major design issue for some of these systems, for the consultants in particular, is whether the system needs to explain its reasoning to a user. This capability is implemented primarily in the effort to convince users that the system's reasoning is appropriate and that its conclusions about a case are reasonable. In some cases, however, the problem-solving expertise used by the system is in a form that is not at all similar to the expertise that an expert user would use to obtain the solution. For example, in the case of the DENDRAL programs, the generator of chemical structure solutions uses a procedure for exhaustively producing solutions based on various graph theoretic notions that the average organic chemist using the system is unlikely to know or care about. Thus a major portion of the DENDRAL expertise resides in a procedure that is conceptually *opaque* to the normal user. The generator was developed because it was discovered that the method used by the chemist to generate solutions is incomplete and the method used by the DENDRAL program has been mathematically proven complete. A similar situation exists in the MACSYMA system, which uses the Risch algorithm for evaluating various types of integrals. While mathematically correct, it is rarely employed by human mathematicians because of its complexity. The correctness and continued success of the programs serve as their primary form of explanation: The user community is thus convinced that the performing system is both acceptable and useable.

In contrast, systems such as MYCIN and PROSPECTOR have been designed to represent and explain the reasoning process used by the system in a manner that is understandable to the knowledgeable user. These systems require a representational formalism capable of supporting the reasoning and explanation abilities that would closely approximate the conceptual structure of expert and user. Since most of these scientific and technical domains have a well-defined set of concepts that their practitioners use consistently, the systems designers have capitalized on this consistency and have designed the programs to accept and reason with knowledge using these concepts.

Assuming a system has an explanation facility, the system designer faces another issue: Should the system reason and apply the expertise in a manner that resembles the methods employed by the human expert? In MYCIN, for example, no claim is made by the designers that the simple backward chaining reasoning methodology has any strong resemblance to the methods actually employed by human physicians performing infectious disease diagnosis. Although the medical concepts employed by the system are familiar to most physicians, the method of inferring the infections and causal organisms, while understandable by a physician, bears little resemblance to a doctor's normal diagnostic reasoning. By contrast, the PIP and INTERNIST systems emphasize the similarities of their diagnostic procedures to those used by physicians.

Knowledge acquisition. During the development of the knowledge base, the expert is unlikely to present all of the relevant facts and relationships that are required for expert performance in the domain. Being human, experts tend to forget or simplify details about their

knowledge, requiring the system to be able to augment its knowledge at a later time. Since the knowledge imparted to the system is largely empirical and the domains are themselves rapidly developing, it is necessary that the system be able to perform these changes easily and in an *incremental* or modular fashion. Thus, most of the recent applications systems have emphasized the use of representation vehicles that allow for the incremental construction of the knowledge base.

Many researchers use *production rules* to perform this incremental construction. Each rule and rule set represents a "chunk" of domain expertise that is communicable to the user and that can be added or extracted with relative ease. Thus the performance of the system can be improved by modifying the knowledge base with new rule sets that deal with new domains or subdomains. Furthermore, the production rule formalism can directly accommodate the concepts of the domain expert and thus is more easily communicable to both the user and the expert.

The Future

A primary research activity in the near future will be the development of facilities for acquiring the domain concepts and the empirical knowledge that these systems require. At present this is a painful process involving many individuals, including both domain experts and computer scientists who together construct the knowledge base. More efficient interfaces for acquiring this domain-specific knowledge, along the lines of the TEIRESIAS system (see article B) and the methods used by the Meta-DENDRAL system (see article C2c), need to be developed before significantly larger expert systems can be constructed.

While the domains and methods that have been developed are interesting and challenging in their own right, they represent only a small fraction of the total cognitive or even conceptually cognitive abilities that a human possesses. These abilities are for the most part as yet undefined in current cognitive research; if they were, they would probably be the subjects of further AI research.

The size of current systems is typically given in terms of some convenient measurement of the domain-specific knowledge contained by the system. For example, the MYCIN system contains approximately 450 rules and a similar number of clinical parameters that it uses to diagnose and prescribe treatments for patients with bacteremia, cystitis, and meningitis. The SYNCHEM system contains approximately 390 transforms that it uses to construct plausible organic synthesis routes. The order of magnitude of expert knowledge has been primarily a function of expert involvement and effort. These systems can potentially support larger knowledge bases but there has been no effort yet to construct these more comprehensive systems. At present, only selected subdomains are actually represented and used.

It is clear that AI and computer science will have to develop new techniques for handling the truly large-scale knowledge bases that will exist in the future. A step in this direction has been taken with the development by Davis (1976) on a representation for knowledge about domain knowledge or *meta-knowledge*. This domain-specific knowledge is used to determine the consistency and appropriateness of various knowledge sources developed and used by the system. The use of meta-knowledge is one of the ways knowledge can be organized both dynamically and statically so that it is comprehensible not only to the machine but also to the human user and expert.

An Application Article

An article on the individual applications systems in this chapter will attempt to cover the following topics:

> A description of the problem domain (e.g., chemistry, infectious disease, etc.), the particular task the application system was designed to perform (e.g., elucidate chemical structures, diagnose and treat a patient with an infectious disease, etc.), and the major motiviations behind the system's design, both for AI and for the task domain.
>
> A description of the task-specific knowledge used by the system to perform the problem-solving task (e.g., knowledge about probable bond breaks for a compound in a mass-spectrometer, knowledge about possible infections and their causal organisms, etc.).
>
> A description of the particular AI methods that were used to represent this knowledge and a description of how the represented knowledge is used to reason about a particular case. This description sometimes includes an annotated sample interaction between a user and the system.
>
> An indication of the current level of expertise of these systems and an indication of their present status and possible future development.

Throughout these articles, emphasis is placed on illuminating the major issues dealt with, and contributions made to Artificial Intelligence by the design of these systems.

References

Feigenbaum (1977) gives a short review of this area of research. The textbook by Winston (1977) also reviews this area briefly. Recent work on some of the important systems is described in a special issue of the Journal of Artificial Intelligence (Sridharan, 1978).

B. TEIRESIAS--Issues in Expert Systems Design

TEIRESIAS is a system for facilitating automatic acquisition and maintenance of the large knowledge bases used by expert systems. Although TEIRESIAS is not itself an application of AI to some domain, it deals with many important issues in expert systems design that are relevant to all of the programs described in this chapter. The system was developed by Randall Davis as part of his doctoral research at the MYCIN project at Stanford, and this article assumes some familiarity with MYCIN's rule-based knowledge representation scheme and its *backward-chaining* control structure (see Article Medicine.C2). However, the ideas and techniques that TEIRESIAS uses are not necessarily limited to MYCIN's domain of infectious diseases or to the production-rule formalism used by MYCIN.

Knowledge-based Programs

As discussed in the Overview, systems that achieve expert-level performance in problem-solving tasks derive their power from a large store of task-specific knowledge. As a result, the creation and management of large knowledge bases and the development of techniques for the informed use of knowledge are now central problems of AI research. TEIRESIAS was written to explore some of the issues involved in solving these problems.

Most expert programs embody the knowledge of one or more experts in a field, like infectious diseases, and are constructed in consultation with these experts. Typically, the computer scientist *mediates* between the experts and the program he is building to model their expertise. This is a difficult and time-consuming task, because the computer scientist must learn the basics of the field in order to ask good questions about what the program is supposed to do.

TEIRESIAS's goal is to reduce the role of the human intermediary in this task of *knowledge acquisition*, by assisting in the construction and modification of the system's database. The human expert communicates, via TEIRESIAS, with the *performance program* (e.g., MYCIN), so that he can discover, with TEIRESIAS's help, what the performance program is doing and why. TEIRESIAS offers facilities for modifying or adding to the knowledge base to correct errors: Using TEIRESIAS, the human expert can "educate" the program just as he would tutor a human novice who makes mistakes. Ideas about how this "debugging" process is best carried out are at the core of TEIRESIAS's success.

TEIRESIAS also recognizes the inexact, experiential character of the knowledge that is often required for knowledge-based systems and (as examples below will illustrate) offers the expert some assistance in formulating new "chunks of knowledge" of this sort. Another major aim of the system was to provide a mechanism for embodying strategic information. *Meta-rules* (discussed below) are used to direct the use of object-level rules in the knowledge base and to provide a mechanism for encoding problem-solving strategies.

Interactive Transfer of Expertise.

It is an established result that an expert knows more about a field than he is aware, or capable of articulating completely. Thus, asking him a broad question like "Tell me everything you know about staph-infections" will yield only a fraction of his knowledge. TEIRESIAS's

approach is to present the expert with some errors made by an already established, but still incomplete, knowledge-based program and to ask a *focused* question: "What do you know that the program doesn't know, which makes your expert diagnosis different in this case?"

This interaction is called *transfer of expertise*: TEIRESIAS incorporates into the performance program the capabilities of the human expert. TEIRESIAS does not attempt to derive new information on its own but, instead, tries to "listen" as attentively and intelligently as possible, to help the expert augment or modify the knowledge base.

Interactive transfer of expertise between an expert and an expert program begins when the expert identifies an error in the performance of the program and invokes TEIRESIAS to help track down and correct the error. Errors are manifest as program responses that the expert would not have made or as "lines of reasoning" that the expert finds odd, superfluous, or otherwise inappropriate. The first kind of error might be, for example, a wrong conclusion about the identity of a bacteria. On the other hand, the performance program may just ask the expert, during a consultation, a question that, in the expert's opinion, does nothing to resolve the identity of the bacteria. This is an example of the "line of reasoning" type of error.

Both kinds of error are assumed, by TEIRESIAS, to be indicative of a deficit, or "bug," in the performance program's knowledge base. Transfer of expertise begins when TEIRESIAS is called upon to correct the deficit. TEIRESIAS fixes bugs in the knowledge base by:

1. Stopping the performance program when the human expert identifies an error.

2. Working backwards through the steps in the performance program that led to the error, until the bug is found.

3. Helping the expert fix the bug by adding or modifying knowledge.

To identify faulty reasoning steps in the performance program, the expert can use the WHY and HOW commands to ask TEIRESIAS to back up through previous steps, *explaining* why they were taken. The same explanatory abilities can also be used when there is no bug, to help the user follow the system's line of reasoning. Since many large performance programs carry out very complex inferences that are essentially "hidden" from the person using the program, this is a valuable facility.

Meta-level Knowledge

One of the principal problems of AI is the question of appropriate representation and use of knowledge about the world (see Representation). Numerous techniques have been used to represent domain knowledge in various applications programs. A central theme of the research on TEIRESIAS is exploring the use of *meta-knowledge*. Meta-level knowledge is simply the representation in the program of knowledge about the program itself--about how much it knows and how it reasons. This knowledge is represented using the same representation techniques used to represent the domain knowledge, yielding a program containing *object-level* representations describing the external world and *meta-level* representations that describe the internal world of the program, its self-knowledge. For example, many AI programs use the notion of a *frame* to represent the knowledge used by the

B. TEIRESIAS--Issues in Expert Systems Design

TEIRESIAS is a system for facilitating automatic acquisition and maintenance of the large knowledge bases used by expert systems. Although TEIRESIAS is not itself an application of AI to some domain, it deals with many important issues in expert systems design that are relevant to all of the programs described in this chapter. The system was developed by Randall Davis as part of his doctoral research at the MYCIN project at Stanford, and this article assumes some familiarity with MYCIN's rule-based knowledge representation scheme and its *backward-chaining* control structure (see Article Medicine.C2). However, the ideas and techniques that TEIRESIAS uses are not necessarily limited to MYCIN's domain of infectious diseases or to the production-rule formalism used by MYCIN.

Knowledge-based Programs

As discussed in the Overview, systems that achieve expert-level performance in problem-solving tasks derive their power from a large store of task-specific knowledge. As a result, the creation and management of large knowledge bases and the development of techniques for the informed use of knowledge are now central problems of AI research. TEIRESIAS was written to explore some of the issues involved in solving these problems.

Most expert programs embody the knowledge of one or more experts in a field, like infectious diseases, and are constructed in consultation with these experts. Typically, the computer scientist *mediates* between the experts and the program he is building to model their expertise. This is a difficult and time-consuming task, because the computer scientist must learn the basics of the field in order to ask good questions about what the program is supposed to do.

TEIRESIAS's goal is to reduce the role of the human intermediary in this task of *knowledge acquisition*, by assisting in the construction and modification of the system's database. The human expert communicates, via TEIRESIAS, with the *performance program* (e.g., MYCIN), so that he can discover, with TEIRESIAS's help, what the performance program is doing and why. TEIRESIAS offers facilities for modifying or adding to the knowledge base to correct errors: Using TEIRESIAS, the human expert can "educate" the program just as he would tutor a human novice who makes mistakes. Ideas about how this "debugging" process is best carried out are at the core of TEIRESIAS's success.

TEIRESIAS also recognizes the inexact, experiential character of the knowledge that is often required for knowledge-based systems and (as examples below will illustrate) offers the expert some assistance in formulating new "chunks of knowledge" of this sort. Another major aim of the system was to provide a mechanism for embodying strategic information. *Meta-rules* (discussed below) are used to direct the use of object-level rules in the knowledge base and to provide a mechanism for encoding problem-solving strategies.

Interactive Transfer of Expertise.

It is an established result that an expert knows more about a field than he is aware, or capable of articulating completely. Thus, asking him a broad question like "Tell me everything you know about staph-infections" will yield only a fraction of his knowledge. TEIRESIAS's

approach is to present the expert with some errors made by an already established, but still incomplete, knowledge-based program and to ask a *focused* question: "What do you know that the program doesn't know, which makes your expert diagnosis different in this case?"

This interaction is called *transfer of expertise*: TEIRESIAS incorporates into the performance program the capabilities of the human expert. TEIRESIAS does not attempt to derive new information on its own but, instead, tries to "listen" as attentively and intelligently as possible, to help the expert augment or modify the knowledge base.

Interactive transfer of expertise between an expert and an expert program begins when the expert identifies an error in the performance of the program and invokes TEIRESIAS to help track down and correct the error. Errors are manifest as program responses that the expert would not have made or as "lines of reasoning" that the expert finds odd, superfluous, or otherwise inappropriate. The first kind of error might be, for example, a wrong conclusion about the identity of a bacteria. On the other hand, the performance program may just ask the expert, during a consultation, a question that, in the expert's opinion, does nothing to resolve the identity of the bacteria. This is an example of the "line of reasoning" type of error.

Both kinds of error are assumed, by TEIRESIAS, to be indicative of a deficit, or "bug," in the performance program's knowledge base. Transfer of expertise begins when TEIRESIAS is called upon to correct the deficit. TEIRESIAS fixes bugs in the knowledge base by:

1. Stopping the performance program when the human expert identifies an error.

2. Working backwards through the steps in the performance program that led to the error, until the bug is found.

3. Helping the expert fix the bug by adding or modifying knowledge.

To identify faulty reasoning steps in the performance program, the expert can use the WHY and HOW commands to ask TEIRESIAS to back up through previous steps, *explaining* why they were taken. The same explanatory abilities can also be used when there is no bug, to help the user follow the system's line of reasoning. Since many large performance programs carry out very complex inferences that are essentially "hidden" from the person using the program, this is a valuable facility.

Meta-level Knowledge

One of the principal problems of AI is the question of appropriate representation and use of knowledge about the world (see Representation). Numerous techniques have been used to represent domain knowledge in various applications programs. A central theme of the research on TEIRESIAS is exploring the use of *meta-knowledge*. Meta-level knowledge is simply the representation in the program of knowledge about the program itself--about how much it knows and how it reasons. This knowledge is represented using the same representation techniques used to represent the domain knowledge, yielding a program containing *object-level* representations describing the external world and *meta-level* representations that describe the internal world of the program, its self-knowledge. For example, many AI programs use the notion of a *frame* to represent the knowledge used by the

system (see Article Representation.C7). One can imagine a meta-level frame that describes the structure of all frames in the system or one that denotes the different classes of frames used in the system. One of TEIRESIAS's representations is very close to this notion, the *schema* described below.

Meta-level knowledge has taken several different forms as its uses have been explored, but it can be summed up as "knowing about what you know." In general, it allows the system both to use its knowledge directly and to examine it, abstract it, and direct its application. The capabilities for explanation, knowledge acquisition, and strategic reasoning in TEIRESIAS inspired the incorporation of explicit meta-level knowledge, and these capabilities are based on the use of that knowledge.

Explanation

There are two important classes of situations where expert systems should be able to explain their behaviour and results. For the user of the system who needs clarification or reassurance about the system's output, the explanation can contribute to the *transparency* and thus the *acceptance* of the system. The second major need for explanation is in the debugging process described above, where a human expert uses the system's explanations of why it has done what it has done, in order to locate some error in the database. The first of these applications of explanation has been explored in the question-answering facility of the MYCIN system; the explanation capability in TEIRESIAS has explored both uses but has concentrated on the latter.

The techniques used in TEIRESIAS for generating explanations are based on two assumptions about the performance program being examined, namely, (a) that a recapitulation of program actions can be an effective explanation, as long as the correct level of detail is chosen, and (2) that there is some shared framework for viewing the program's actions that will make them comprehensible to the user. In the MYCIN-like expert systems that use production-rule knowledge bases, these assumptions are valid, but it is easy to imagine expert systems where one or both are violated. For example, the first assumption simplifies the explanation task considerably, since it means that the solution requires only the ability to record and play back a history of events. This assumption rules out, in particular, any need to simplify those events. However, it is not obvious, for instance, that an appropriate level of detail can always be found. Furthermore, it is not obvious how this approach of recapitulation, which often offers an easily understood explanation in programs that reason symbolically, would be applied to expert systems that perform primarily numeric computations.

A simple recapitulation will be an effective explanation only if the level of descriptive detail is constrained. It must be *detailed* enough that the operations the system cites are comprehensible; the conceptual level must be *high* enough that the operations are meaningful to the observer, so that unnecessary detail is suppressed; and it must be *complete* enough so that the operations cited are sufficient to account for all behavior.

The second assumption concerns the user's comprehension of the expert system's activity, which depends on the fundamental mechanism used by the program and the level at which it is examined. Consider a program that does medical diagnosis using a statistical approach based on Bayes's Theorem. It is difficult to imagine what explanation of its actions

the program could give if it were queried about computed probabilities. No matter what level of detail is chosen, such a program's actions are not (nor were they intended to be) a model of the reasoning process typically employed by physicians. Although they may be an effective way for the computer to solve the diagnosis problems, there is no easy way to interpret these actions in terms that will make them comprehensible to humans unacquainted with the program.

Thus, the lack of mechanisms for simplifying or reinterpreting computation means that TEIRESIAS's approach is basically a first-order solution to the general problem of explanation. But, in the context of a MYCIN-like expert system, for which TEIRESIAS was designed, the simple AND/OR goal tree control structure offers a basis for explanations that typically needs little additional clarification. (The operation of TEIRESIAS's explanation facility is illustrated in the sample protocol at the end of this article.) The invocation of a rule is taken as the fundamental action of the system. This action, within the framework of the goal tree, accounts for enough of the system's operation to make a recapitulation of such actions an acceptable explanation. In terms of the constraints noted earlier, it is sufficiently detailed--the actions performed by a rule in making a conclusion, for instance, correspond closely enough to the normal connotation of that word--that no more detailed explanation is necessary. The explanation is still at a high enough conceptual level that the operations are meaningful and the explanation is complete enough--there are no other mechanisms or sources of information that the observer needs to know in order to understand how the program reached its conclusions.

Knowledge-acquisition: Rule Models and Schemata

When the expert has identified a deficit in the knowledge base of the performance program, TEIRESIAS questions him in order to correct the deficit. This process relies heavily on meta-level knowledge about the performance program, encoded in *rule-models* and *schemata*. In other words, TEIRESIAS knows about what the performance program knows.

The meta-level knowledge about *objects* in the domain includes both structural and organizational information and is specified in *data structure schemata*. Acquisition of knowledge about new objects proceeds as a process of instantiating a schema--creating the required structural components to build the new data structure and then attending to its interrelations with other data structures. By making inquiries in a simple form of English about the values of the schema's components, this knowledge acquisition process is made to appear to the expert as a natural, high-level inquiry about the new concept. The process is, of course, more complex, but the key component is the system's description of its own representation.

TEIRESIAS's *rule models* are empirical generalizations of subsets of rules, indicating commonalities among the rules in that subset. For example, in MYCIN there is a rule model for the subset of rules that conclude affirmatively about *organism category*, indicating that most such rules mention the concepts of *culture site* and *infection type* in their premise. Another rule model notes that those rules that mention *site* and infection *type* in the premise also tend to mention the *portal of entry* of the organism.

This knowledge about the contents of the domain rules is used by TEIRESIAS to build *expectations* about the dialogue. These expectations are used to facilitate the process of translating the English statements into the performance program's internal representation

and to identify information missing from the expert's entry. An example of TEIRESIAS's use of rule models in its knowledge acquisition dialogue is given in the sample protocol below.

Meta-rules and Performance Strategies

In performance programs with sufficiently small knowledge bases (like MYCIN's), exhaustive invocation of the relevant parts of the knowledge base during a consultation is still computationally feasible. In time, however, with the inevitable construction of larger knowledge bases, exhaustive invocation will prove too slow. In anticipation of this eventuality, *meta-rules* are implemented in TEIRESIAS as a means of encoding strategies that can direct the program's actions more selectively than can exhaustive invocation. The following meta-rule is from MYCIN's infectious disease domain:

METARULE 001

If 1) the infection is a pelvic-abscess, and
2) there are rules which mention in their premise enterobacteriaceae, and
3) there are rules which mention in their premise gram positive rods,

Then There is suggestive evidence (.4) that the rules dealing with enterobacteriaceae should be evoked before those dealing with gram positive rods.

This rule suggests that since enterobacteriaceae are commonly associated with a pelvic abscess, it is a good idea to try rules about them first, before the less likely rules mentioning gram positive rods. Note that this meta-rule does not refer to specific object-level rules. Instead it specifies certain attributes of the rules it refers to, for example, that they mention in their premise enterobacteriaceae.

An Example: TEIRESIAS in the Context of MYCIN

We will now illustrate TEIRESIAS's operation in affiliation with the MYCIN system (see Article Medicine.C2), paying particular attention to TEIRESIAS's explanation and knowledge acquisition facilities. MYCIN provides the physician with advice about the diagnosis and drug therapy for bacterial infections. The system asks questions about the patient, the infection, the cultures grown from specimens from the patient, and any organisms (bacterium) growing in the culture. (Typically, of course, the exact identity of the organism is not yet known.)

MYCIN's database is composed of rules that specify a situation (involving information about the patient, culture, and organism) and the conclusions that can be drawn in that situation. For example, to conclude whether a patient suffers from a bacteria in the enterobacteriaceae category, MYCIN invokes Rule 95 of its knowledge base:

Rule 095

IF the site of the culture is blood, and
the gram stain is positive, and
the portal of entry is gastrointestinal tract, and
[A-the abdomen is the locus of infection, or
B-the pelvis is the locus of infection]

THEN there is strongly suggestive evidence that
enterobacteriaceae is the class of organisms for
which therapy should cover.

MYCIN asks questions of the physician to determine which rules are applicable in the current case--i.e., to determine the "situation." To determine if the organism is of type enterobacteriaceae, MYCIN would inquire whether each clause in Rule 95 were true.

The procedure of working backwards from the conclusion to the situation that would make the conclusion true is called *backward chaining* and is MYCIN's fundamental control structure. Backward chaining can go "several levels deep," meaning that the clauses in a situation that must be true (for the conclusion to be true) might themselves be conclusions of other rules. For example, a condition on the identification of enterobacteriaceae is that the infection be localized in the abdomen or the pelvis; but this condition is itself a conclusion, and it is only true if its conditions (that the culture from these places is recent, and is therapeutically significant) are true. Furthermore, the condition of therapeutic significance has its own conditions, and so on. Here is a protocol showing MYCIN's basic style of interaction, the human expert's responses are prefixed with **:

---------Patient 281----------
1) Patient's name:
** Albert Hall
2) Sex:
** Male
3) Age:
** 34
4) Have you been able to obtain positive microbiological
information about a possible infection of Albert Hall?
** Yes

---------Infection 1-----------
5) What is the infection?
** Primary-Bacteremia
6) Please state when signs and symptoms of infection-1 first appeared:
** 15-Dec-75 12:00
---------Culture 1-------------
7) From what site was the specimen for culture-1 taken
** Blood
8) Please give the date and time when this blood culture was obtained:
** 16-Dec-75 12:00
The first significant organism from this blood culture will be

```
referred to as:
---------Organism-1----------
9) Enter the identity of ORGANISM-1
** Unknown
10) Is ORGANISM-1 a rod, or coccus, (etc.):
** Rod
11) The gram-stain of ORGANISM-1:
** Gram negative
12) Did ORGANISM-1 grow in the aerobic bottle?
** Yes
13) Did ORGANISM-1 grow in the anaerobic bottle?
** Yes
                    :
                    :
                    :
My therapy recommendations will be based on the following probable
infection(s) and potential causative organism(s):

INFECTION-1 is PRIMARY-BACTEREMIA
      <item-1> E.COLI [ORGANISM-1]
      <item-2> PSEUDOMONAS-AERUGINOSA [ORGANISM-1]
```

FIGURE 1. PART OF A DIALOGUE WITH MYCIN.

In sum, MYCIN constructs a *goal tree* (see Article Search.A2) of questions that must be resolved in the course of the consultation to conclude the identity of a bacteria. When it cannot resolve a question by inference from what it knows already, it asks the respondent to provide an answer. Each node of the goal tree has subnodes that result from the application of a rule:

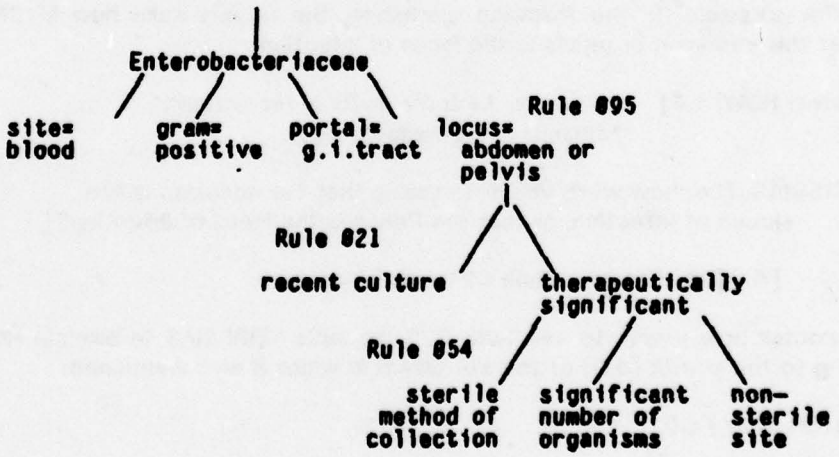


Figure 1. A MYCIN Goal Tree

Explanation: HOW and WHY. Suppose that during a consultation with MYCIN the clinician wants an explanation of the reasoning behind some question that MYCIN has asked him. The explanation given by TEIRESIAS in response to the physician's question, WHY, involves backing up the goal tree to the previous node and presenting the rule used to derive the subnode from its ancestor:

MYCIN: What is the suspected portal of entry of ORGANISM-1 into this sterile site?
Doctor:WHY
TEIRESIAS: [i.e. why is it important to determine the suspected portal of entry of ORGANISM-1 into this sterile site?]

[3.0]...in order to find out about an organism or class of organisms for which therapy should cover.

It has already been established that
[3.1] the site of the culture is blood
[3.2] the stain of the organism is gram positive
Therefore if:
[3.3] the portal of entry of the organism is g.i.
[3.4] [Abdomen is the locus of infection, or Pelvis is the locus of infection]
Then
There is strongly suggestive evidence (.9) that enterobacteriaceae is the class of organisms for which therapy should cover.
[Rule 095]

In contrast to WHY, which spells out "higher" nodes in the goal tree, the HOW command asks TEIRESIAS to explain the nodes that have been or will be expanded lower down the tree. For example, in the following exchange, the expert asks how MYCIN will determine whether the abdomen or pelvis is the locus of infection:

Doctor: HOW[3.4] *note: 3.4 is the prefix of the statement of interest in the exchange above*

TEIRESIAS: [i.e. how were you determining that the Abdomen is the locus of infection, or that the Pelvis is the locus of infection?]

[4.0] At that point Rule 021 was being used.

If the doctor now wants to see Rule 021, he asks TEIRESIAS to explain HOW it was used, referring to the prefix (4.0) of the statement in which it was mentioned:

Doctor: HOW [4.0]

TEIRESIAS: [i.e. how was Rule 021 used?]

[5.0] It has already been established that
[5.1] the culture is recent.

Therefore if:
[5.2] there is therapeutically significant disease associated with the occurrence of this organism
then
it is definite (1.0) that the site of the culture is the locus of infection in the patient.

The explanation facilities of TEIRESIAS involve climbing up and down MYCIN's goal tree explaining WHY questions are asked and HOW they are answered. If MYCIN makes a misdiagnosis, these same facilities are used (either by the expert or TEIRESIAS, at the choice of the expert) to back up through the rules that achieved the diagnosis until the expert has found the faulty one or determined that one is missing. TEIRESIAS is then used to expedite the alteration or construction of rules by the expert.

Knowledge acquisition. TEIRESIAS is able to help an expert formulate rules in English because it has expectations about what rules "look like." The expectations are stored internally in the form of *rule models* that are formed automatically by examining all the rules in the knowledge base. TEIRESIAS knows, for example, that there is a class of rules that have as their conclusions the probable identity of a bacteria and that these rules typically have as conditions (a) the site of the culture, (b) the stain of the culture, and (c) the probable portal of entry of the bacteria. These rule models help TEIRESIAS translate English versions of new rules into a machine representation acceptable to MYCIN.

To illustrate TEIRESIAS's debugging and correction facilities, we will continue with the consultation from Figure 1. Suppose that after MYCIN presents the expert with its diagnosis, TEIRESIAS steps in and checks whether the expert agrees:

MYCIN: My therapy recommendation will be based on the following probable infection(s) and potential causative organism(s):

INFECTION-1 is PRIMARY BACTEREMIA
<Item-1> E.COLI [ORGANISM-1]
<Item-2> PSEUDOMONAS-AERUGINOSA [ORGANISM-1]

Now TEIRESIAS steps in.

Are all these identities correct?

*The expert's response, marked with **, is No.*

** No.
Are there any identities missing?
** Yes.
Which ones?
** KLEBSIELLA-PNEUMONIAE, PROTEUS-NON-MIRABILIS
Are there any identities which shouldn't have appeared?
** yes

Which one(s)?
** PSUEDOMONAS-AERUGINOSA

Do you want to
1] track down the problem yourself, or
2] allow me to guide the debugging, or
3] ignore it for now and fix it another time.
Type 1,2, or 3.

** 2

If the expert had typed 1, he would have used the HOW and WHY commands to debug the problem, as discussed above.

We'll start with PSEUDOMONAS-AERUGINOSA first:

TEIRESIAS starts debugging by trying to find out why an inappropriate diagnosis occurred. Frequently solving a problem will make others "go away" as well.

The following rules were used in deducing that the identity of ORGANISM-1 is PSEUDOMONAS-AERUGINOSA:

Rule 184

Since
[1.1] the category of ORGANISM-1 is not known
[1.2] the gram stain of ORGANISM-1 is gram negative
[1.3] the morphology of ORGANISM-1 is rod
[1.4] the aerobicity of ORGANISM-1 is facultative

There is weakly suggestive evidence (.3) that the identity of ORGANISM-1 is PSEUDOMONAS-AERUGINOSA.

Is this rule correct

Note that this rule, which resulted in the wrong conclusion, is not necessarily an incorrect rule. It might be (and in fact is) the case that it was incorrectly invoked because one or more of its conditions are true when they should be false.

** Yes

The expert says the rule is correct.

Should its premise have failed for this case?
** Yes

Which clause of the premise should have failed?
**** Clause 1.**

The expert believes that MYCIN should have known the category of ORGANISM-1. MYCIN failed to establish the category, causing Rule 184 to be used.

Should it have been possible to conclude a category?
**** Yes**
Which one?
**** Enterobacteriaceae**

Our old friend. But recall that MYCIN's rule 095 about enterobacteriaceae requires a positive gram stain, but it can be seen from Fig. 1 that it has a negative gram stain. Thus, Rule 95 was never evoked. In the course of more debugging, it transpires that MYCIN has a couple other rules dealing with this category of bacteria, but they weren't evoked either. The expert decides that another rule is required, and we pick up the dialogue where he is proposing one to TEIRESIAS:

.
.
.

If: 1 ** THE PATIENT'S INFECTION IS PRIMARY-BACTEREMIA
2 ** THE SITE IS ONE OF THE STERILE SITES
Then: ** THERE IS EVIDENCE (.8) THAT THE CATEGORY IS ENTEROBACTERIACEAE

TEIRESIAS parses this rule into its internal representation and then translates it back into English, so the expert can check it. We pick up the dialogue after the expert has agreed on the parse. TEIRESIAS is applying its rule model.

.
.
.

I hate to criticize, Dr. Davis, but did you know that most rules about what the category of an organism might be, that mention-
the site of a culture
the infection
ALSO mention-
[A]- the portal of entry of the organism
Shall I try to write a clause to account for [A]?
**** Yes.**

Here we see one of the utilities of the rule model. TEIRESIAS prompts the expert to add a bit of knowledge that is typcially found in rules like these and offers to fix the rule itself. This completion is accomplished by looking at other rules that fit the same rule model, to find the most likely portal of entry clause.

how about -
[A] The portal of entry is gastrointestinal. Ok?
** Yes.

TEIRESIAS now does some finishing up: checking the complete rule with the expert for final approval and asking the expert to write a brief description (for bookkeeping purposes) of why the rule was needed. Finally, it reruns the consultation internally, using the responses from Fig. 1, which it has stored. It turns out that adding the rule above did, in fact, cure the other problems with the first consultation, and this time the diagnosis is satisfactory to the expert.

.
.
.

Summary: TEIRESIAS and Expert Systems

TEIRESIAS aids a human expert in monitoring the performance of a knowledge-based system. When the human expert spots an error in the program's performance, either in the program's conclusions or its "line of reasoning," TEIRESIAS assists in finding the source of the error in the database by *explaining* the program's conclusions--retracing the reasoning steps until the faulty (or missing) rule is identified. At this point, TEIRESIAS assists in *knowledge acquisition*, modifying faulty rules or adding new rules to the database. *Meta-level knowledge* about the kinds of rules and concepts in the database is used to build expectations in TEIRESIAS's *model-based understanding* process. Meta-level knowledge is also used to encode problem-solving *strategies*, in particular, to order the invocation of rules so that those that are most likely to be useful (given the current knowledge of the program) are tried first.

References

The principal reference on TEIRESIAS is the doctoral dissertation by Davis (1976). Uses of meta-knowledge in expert systems are discussed in Davis and Buchanan (1977). Also see Davis (1977) and Davis (1978).

C. Applications in Chemistry

C1. Chemical Analysis

Computer programs have been developed to aid in almost every aspect of chemistry. As evidenced by recent articles in two journals devoted to uses of computers for chemical problems, **Computers and Chemistry** and **Journal of Chemical Information and Computer Science**, most of the computer programs have focused on numeric problems of data acquisition, data reduction, complex electronic energy calculations, and the like. By contrast, AI methods have found application in two major classes of nonnumeric chemical reasoning problems: (a) determining the molecular structure of an unknown organic compound, the "analysis" or "structure determination" problems; and (b) planning a sequence of reactions in order to synthesize organic chemical compounds, the "synthesis" problems (see Article C4).

Structure Elucidation

The elucidation of molecular structures is fundamental to the application of chemical knowledge to important problems in biology and medicine. Some of the areas in which chemists maintain active interest include: (a) identification of naturally occurring chemical compounds isolated from terrestrial or marine organisms; (b) verification of the identity of new synthetic materials; (c) identification of drugs and their metabolites in clinical studies; and (d) detection of metabolic disorders of genetic, developmental, toxic, or infectious origins through the identification of organic constituents excreted in abnormal quantities in human body fluids.

In many circumstances, especially in the areas of interest mentioned above, the powerful techniques of x-ray crystallography and x-ray fine-structure analysis may not be applicable (see article C3), and chemists must resort to structure elucidation based on data obtained from a variety of other methods. Foremost among them historically is mass spectrometry (discussed in detail in the next section). If a chemist wants to determine the molecular structure of an unknown chemical compound, he first isolates a sample of the compound that is pure--i.e., contains no other compounds. Two questions must then be answered:

1. What are the atoms in the compound?
2. How are the atoms arranged (joined together) in a three-dimensional structure?

The latter question is addressed by structure elucidation programs. It is relatively simple to determine the constituents of the molecule (the first question), but the enormous number of possible three-dimesnional arrangements makes the second question especially difficult to answer. If the unknown substance is a crystal, or can be crystalized, then x-ray crystallography can be used to determine the exact locations and connections of atoms in a molecule in space. If this technique cannot be used and x-ray fine-structure analysis techniques cannot be applied, then the chemist must take a more complicated approach to structure elucidation. No other tests are available to tell the chemist the *exact* structure of his molecule; at best he can use tests that help him discover small connected clusters of

atoms, called *molecular fragments*, which are either present or absent in the compound. Therefore, although the chemist may not know the structure of the molecule, he does know some of its subparts. From the fragments identified as present in the compound and those known to be absent, the chemist can derive a set of *constraints*. A constraint can be thought of as a piece of a graph that either must occur or must not occur in the final graph of the molecule. This is how constraints are represented in the structure elucidation programs that we will discuss.

Using the known constraints about a given molecule, it is often possible to generate the graphs of all molecules that adhere to those constraints. An algorithm was developed by Lederberg (1964) to generate all possible acyclic molecular structures from a set of atoms; and Brown, et al. (1974) developed an algorithm without the acyclic constraint. Thus it is now theoretically possible to generate every possible molecular structure containing known subparts, but it is often prohibitively expensive to do so. However, the exhaustive generation algorithm can often be constrained to produce a relatively small set of molecular structures, one of which is the unknown molecule.

If the number of atoms in an unknown molecule is relatively small and the number of known constraints is large, a chemist can figure out the molecular structure by hand. However, the manual approach has been significantly augmented by computer programs developed in the DENDRAL project at Stanford University. These programs do not generate all the possible moelcular structures and then discard structures according to the constraints; rather, they use the constraints to insure that only a small subset of the theoretically possible structures are ever actually generated.

Structure Elucidation with Constraints from Mass Spectrometry

As we mentioned above, structure elucidation programs are designed to help organic chemists determine the molecular structure of unknown compounds. Experimental data from the unknown may be gathered from many different analytic techniques including mass spectrometry (MS), nuclear magnetic resonance spectroscopy (NMR), infrared spectroscopy (IR), ultraviolet spectroscopy (UV), and "wet chemistry" analysis. Mass spectrometry is still a new and developing technique. It is particularly useful when the quantity of the sample to be identified is very small; mass spectrometry requires only micrograms of sample.

A mass spectrometer bombards the chemical sample with electrons, causing *fragmentations* and rearrangements of the molecules. Charged fragments are collected by mass. The data from the instrument, recorded in a histogram known as a mass spectrum, show the masses of charged fragments plotted against the relative abundance of the fragments at a given mass. Although the mass spectrum for each molecule may be nearly unique, it is still a difficult task to infer the molecular structure from the 100-300 data points in the mass spectrum; not only does a spectrum contain "noise peaks" and overlapping peaks originating from many parts of the molecule, but the theory of mass spectrometry is not complete.

Throughout this section the following terms will be used to describe the actions of molecules in the mass spectrometer:

Fragmentation--the breaking of a connected graph (molecule) into fragments by breaking one or more edges (bonds) within the graph.

Atom migration--the detachment of nodes (atoms) from one fragment and their reattachment to other fragments. This process alters the masses of all of the fragments.

Mass spectral process--a fragmentation followed by zero or more atom migrations.

Other analytic techniques are commonly used in conjunction with, or instead of, mass spectrometry. Some rudimentary capabilities exist in structure elucidation programs to interpret proton NMR and Carbon 13 (13C) NMR spectra. For the most part, however, interpretation of other spectroscopic and chemical data has been left to the chemist. The programs still need the capability to integrate the chemist's partial knowledge into the generation of structural alternatives.

We will now consider two programs that utilize mass spectrometry constraints in the elucidation of organic compound structures: DENDRAL and Meta-DENDRAL.

C2. The DENDRAL Programs

C2a. DENDRAL

In 1964 Joshua Lederberg developed the DENDRAL algorithm, which produces all possible acyclic (unringed) molecular structures, given a set of atoms. This algorithm enabled an exhaustive approach to structure elucidation. In 1965 the DENDRAL project started at Stanford. One intent of the project was to show that algorithmic programs that produce results exhaustively and at enormous expense could be augmented by some of the heuristic knowledge used by experts to produce much the same results with a fraction of the effort. The Heuristic DENDRAL Program achieved this objective by augmenting the DENDRAL algorithm with a set of rules, those used by expert chemists to infer constraints on molecular structures from mass spectrographic information about the molecule. Unfortunately, pressing expert chemists to formulate rules about mass spectrometry was an arduous process. The theory of mass spectrometry was incomplete, and the rules about it were inexact and heuristic. In 1970, the Meta-DENDRAL project addressed the problem of inferring the rules of mass spectrometry from two sources of information: molecular structures, and their mass spectra. Meta-DENDRAL is a continuing project.

In 1976, the CONGEN program became the center of attention in the DENDRAL project. This program replaced Lederberg's original Heuristic DENDRAL acyclic structure generator with a generator without its limitation. CONGEN is discussed in a separate article (C2b) because it has been used as a stand-alone system by research chemists.

DENDRAL

The Heuristic DENDRAL program was designed to find a relatively small set of possible molecular structures, given the atoms in the molecule and the mass spectrum of the molecule. The limitations of the DENDRAL algorithm were such that Heuristic DENDRAL could generate only acyclic (unringed) structures: Ketones, alcohols, ethers, thiols, thioethers, and amines.

The program has three functional parts: Plan, generate, and test.

1. PLAN: Planning in this context means redefining the problem in terms that will reduce the effort of the problem solver--e.g.,redefine the problem of finding all possible combinations of a set of atoms to the problem of finding all such combinations consistent with constraints derived from mass-spectrometry. Automatic inference of these constraints is the planning part of Heuristic DENDRAL. The list of constraints has two parts: a list of molecular fragments (clusters of atoms) that must be in the final molecular structure and a list of fragments that are forbidden to appear in the final structure.

2. GENERATE: This part uses these constraints to prevent the DENDRAL algorithm from generating structures that include forbidden subparts or that exclude mandatory subparts. The generator was originally derived from Lederberg's algorithm. When CONGEN was implemented as a stand-alone system, these constraints were provided by the chemists using the program, not by the planning part.

3. TEST: This part ranks the resulting list of candidate structures by simulating its

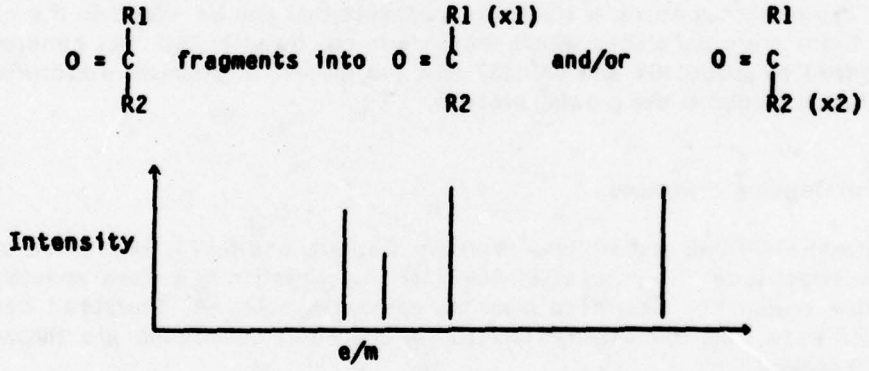
behavior in a mass spectrometer. The structures resulting in simulated spectra close to the empirical one are ranked high.

Heuristic DENDRAL thus has *two* sets of rules that encode the mass spectrometry knowledge: a) rules used during planning that interpret mass spectral data and infer molecular fragments, and, b) rules used during testing that simulate the action of the mass spectrometer on the structure(s) proposed by CONGEN and that predict peaks which should be observed in the spectrum of the molecule.

Planning: Inferring Constraints From The Mass Spectrum

Heuristic DENDRAL has available to it the mass spectrum and the atomic constituents of a molecule. From the latter it can infer the molecular weight, M, of the molecule. Many of the rules for interpreting mass spectra include M; for example, the following rule:

If the spectrum for the molecule has 2 peaks at masses
x1 and x2 such that
a. x1 + x2 = M + 28, and
b. x1 - 28 is a high peak, and
c. x2 - 28 is a high peak, and
d. at least one of x1 or x2 is high, and
Then the molecule contains a ketone group.



This piece of knowledge about mass spectrometry allows Heuristic DENDRAL to constrain its structure-generating algorithm to produce molecules with a ketone group as a mandatory constituent. This rule, in addition to many similar rules, significantly constrains the number of molecules generated by the structure generator. For example, given the spectrum for a molecule containing 8 carbons, 16 hydrogens, and 1 oxygen, the constraint-generating program can eliminate from consideration (i.e., place on a list of forbidden structures called BADLIST) all possible structures except those containing ethyl ketone 3, which reduces the number of generated molecular structures from the topologically possible 790 to a constrained set of 3 (called the GOODLIST).

The Generator

The algorithm for generating molecular structures is complicated and has no AI content; we will discuss it only in general terms and refer the reader to Buchanan, Sutherland, Feigenbaum, 1969, for a detailed discussion. The following article (C2b) discusses the current CONGEN generator.

There are several design characteristics of the generator that are related to the enormous number of molecules combinatorially possible in an analysis problem. First, the generator must be *proved* to be complete--it must be able to generate all topologically possible molecular structures. It should also be non redundant, that is, it should generate each structure only once. Redundancy was a problem for structures with rings, because Lederberg's algorithm treated symmetrical molecules as unique structures. A third characteristic is that the generator should be flexible enough to be focused by constraints from the planning part. It should not blindly generate all possible structures, but only those fulfilling the constraints. If GOODLIST and BADLIST are empty, it should generate all *isomers* (structural variants) of the given composition.

Some simple checks are made by the generator. The composition should be compatible with the constraints inferred from the spectrum, and the structures generated should have only the types and amounts of atoms specified in the composition. Finally, the generator should not produce a structure known by DENDRAL to be unstable.

The structure generator essentially "grows" molecules, starting with a small fragment of the molecule and adding pieces of the composition to it. At any point in the growing process, there are numerous atoms or molecular fragments that can be added to the growing structure, and there are many places where these parts can be attached. But generally the constraints offered by GOODLIST and BADLIST limit the number of possible structures that might be grown at any point in the growing process.

The Testing and Ranking Programs

The programs MSPRUNE and MSRANK (Varkony, Carhart, and Smith, 1977) use a large amount of knowledge about the process of molecular fragmentation in a mass spectrometer to make testable predictions from each plausible candidate molecule. Predicted data are compared to the data from the unknown compound, and some candidates are thrown out, while others are ranked.

MSPRUNE works with: (a) a list of candidate structures from the structure generator, and (b) the mass spectrum of the unknown molecule. It uses a fairly simple model of mass spectrometry (encoded in rules) to predict commonly expected fragmentations for each candidate structure. Predictions that deviate greatly from the observed spectrum are considered *prima facie* evidence of incorrectness, and the corresponding structures are pruned from the list. MSRANK then uses more subtle rules of mass spectrometry to rank the remaining structures according to the number of predicted peaks found (and not found) in the observed data, weighted by measures of importance of the processes producing those peaks.

Research Results

The Heuristic DENDRAL project, from 1966 to the present, and including CONGEN, has produced a number of results of significance to chemists. The effort has shown that it is possible to write a computer program that equals the performance of experts in some very specialized areas of science. Published papers on the program's analysis of aliphatic ketones, amines, ethers, alcohols, thiols, and thioethers (Duffield et al., 1969; Schroll et al., 1969; Buchs et al., 1970) make the point that although the program does not know more than an expert (and in fact knows far less), it performs well because of its systematic search through the space of possibilities and its systematic use of what it does know. A paper on the program's analysis of estrogenic steroids notes that the program can solve structure elucidation problems for complex organic molecules (Smith et al., 1972). Another paper, on the analysis of mass spectra of mixtures of estrogenic steroids (without prior purification), establishes the program's ability to do better than experts on some problems (Smith et al., 1973). With mixtures, the program succeeds where people fail; the task of correlating data points with each possible fragmentation of each possible component of the mixture is too difficult for people to do. Several articles based on results from CONGEN demonstrate its power and utility for solving problems of medical and biochemical importance (Smith, 1975; Smith and Carhart, 1976; Buchanan, 1976; Mitchell, 1976; and Varkony, Carhart, and Smith, 1977).

DENDRAL programs have been used to aid in structure determination problems of the following kinds:

terpenoid natural products from plant and marine animal sources,
marine sterols,
organic acids in human urine and other body fluids,
photochemical rearrangement products,
impurities in manufactured chemicals,
conjugates of pesticides with sugars and amino acids,
antibiotics,
metabolites of microorganisms, and
insect hormones and pheremones.

CONGEN (discussed next) has also been applied to published structure elucidation problems by students in organic chemistry classes to check the accuracy and completeness of published solutions. In several cases, the program found structures that were plausible alternatives to the published structures (based on problem constraints that appeared in the article). This kind of information served as a valuable check on conclusions drawn from experimental data.

References

See Lindsay, Buchanan, Feigenbaum, and Lederberg (forthcoming) for a thorough and current treatment of the DENDRAL programs. Buchanan and Feigenbaum (1978) is a recent, short description of the programs. Also see Buchanan, Sutherland, and Feigenbaum (1969) and Lederberg (1964a).

C2b. CONGEN and its Extensions

CONGEN: Interpretation of Constraints

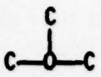
CONGEN (for CONstrained GENerator) is a program that was designed in 1976 to replace the old DENDRAL generator of acyclic structures. It has proved a powerful stand-alone program to assist the chemist in determining the molecular structure of unknown compounds. Its objective was twofold: (a) to allow the user to interactively specify certain types of structural information determined from any of several sources (e.g., spectroscopy, chemical degradation, method of isolation, etc.); and (b) to generate an exhaustive and nonredundant list of structures consistent with this information. Unlike the original Heuristic DENDRAL program, it does not infer *constraints* from mass spectra, but allows the chemist to specify them. Another difference between CONGEN and Heuristic DENDRAL is that the former can generate *cyclic* as well as *acyclic molecular structures*. The generation is a stepwise process, and the program allows interaction at every stage. Based upon partial results, the chemist may be reminded of additional information that he can specify, thus limiting further the number of structural possibilities.

CONGEN breaks down the problem statement given by the chemist in several different ways, for example: (a) hydrogen atoms are omitted until the final steps of processing; (b) parts of the graph containing no cycles are generated separately from cyclic parts (and combined at the end); (c) cycles containing only unlabeled nodes are generated before the nodes are labeled with the names of chemical atoms (e.g., carbon or nitrogen); and (d) cycles containing only three-connected nodes (e.g., nitrogen or tertiary carbon) are generated before two-connected nodes (e.g., oxygen or secondary carbon) are mapped onto the edges. At each step, several constraints may be applied to limit the number of emerging chemical graphs (Carhart et al., 1975).

There are two algorithms at the heart of CONGEN whose validity producing nonredundant structures has been mathematically proven (Brown & Masinter, 1974; Masinter et al., 1974) and whose computer implementation has been well tested. Combined, they are designed to determine all topologically unique ways of assembling a given set of atoms, each with an associated valence, into molecular structures. The atoms may be chemical atoms with standard chemical valences, or they may be names representing *molecular fragments* (*superatoms*) of any desired complexity, where the valence corresponds to the total number of bonding sites available within the superatom. The algorithms can be thought of as performing *problem reduction*, and *reconstruction* or *subproblem recomposition* on molecular structures. The first, *partitioning*, algorithm breaks down the problem of finding a complete molecular structure into subproblems; for example, to find the structures of the ringed and non-ringed components of the molecule. The second, *embedding*, algorithm combines the substructures, found by partitioning, into complete molecular structures. Clearly, neither partitioning nor reconstruction can be unconstrained processes because of the combinatorics involved: There are simply too many possible subproblems to solve, and each of them may have many solutions. Consequently, combining subproblem solutions exhaustively is not feasible. In both algorithms, constraints are brought to bear to limit the size of the problem. Three types of constraints are:

1. Graph theoretic: Symmetric structures are not considered unique.

2. Syntactic: Structures are constrained by the valences of the constituent atoms; for example,



is impossible because oxygen is bivalent, i.e. has only two bonding sites.

3. <Semantic>: The chemist provides additional information about the molecule that will help to determine its structure.

Substantial effort has been devoted to modifying the two basic procedures, particularly the structure generation algorithm--allowing it to accept a variety of other structural information (constraints) and using it to prune the list of structural possibilities. Current capabilities include specification of good and bad substructural features, good and bad ring sizes, proton distributions and connectivities of isoprene units (Carhart and Smith, 1976). Usually the chemist has additional information (if only some general rules about chemical stability) of which the program has little knowledge, which he can use to limit the number of structural possibilities. For example, he may know that the chemical procedures used to isolate the compound would change organic acids to esters; thus, the program would not need to consider structures with unchanged acid groups. In CONGEN, he is given the facilities to impart this knowledge interactively to the program.

To make CONGEN easy for research chemists to use, the program has been provided with an interactive "front end." This interface contains EDITSTRUC, an interactive structure editor; DRAW, a teletype-oriented structure display program; and the CONGEN "executive" program, which ties together the individual subprograms and aids the user with various tasks such as defining superatoms and substructures, creating and editing lists of constraints or superatoms, and saving and restoring superatoms, constraints, and structures from secondary storage (disc). Recently CONGEN was rewritten to search *depth first* so that examples could be produced right from the beginning of the computation. This often allows the chemist to see that a particular problem has been poorly or incorrectly constrained and to stop the computation early, saving large amounts of expensive computer time.

The current system is running on the SUMEX computing facility at Stanford and is available nationwide over the TYMNET network. It has recently been completely re-written in the BCPL programming language to run on a variety of other machines.

Limitations and Extensions

Although computer programs, including CONGEN, now exist to assist chemists in constructing structural isomers based on information about partial structures, the programs have one serious, common limitation. Each program must use non-overlapping structural fragments as building blocks. This limitation leads to at least two important problems. First, the chemist using such a program must select non-overlapping partial structures; otherwise an incomplete set of structures will result. This procedure, done manually, is time-consuming

and prone to error. Second, as a consequence of the first step, problems are solved less efficiently by the programs because a detailed environment of fewer atoms has been specified--to ensure the absence of overlaps.

The GOODLIST INTERPRETER is a first attempt to remove this limitation by simulating the manual procedure that the chemist uses to arrive at a set of non-overlapping constraints. It is designed to make more efficient use of information about required (GOODLIST plus superatoms) structural features of an unknown. Some early successes have demonstrated that new problems are brought within the realm of solution by the GOODLIST INTERPRETER that are impossible in CONGEN alone, due to the constraints on computational resources.

Stereochemistry

One of the most important new additions to CONGEN deals with the problem of enumerating all the stereoisomers of a given compound.

The mathematical problem of enumerating stereoisomers was solved by Jim Nourse. Considerations of symmetry as embodied in the mathematical theory of groups played a decisive role in the solution. Coupled with the stereoisomer generator, and given an empirical formula and a number of constraints, CONGEN can generate all the stereoisomers that are possible solutions to the unknown target molecule to be elucidated.

While the solution to the enumeration of stereoisomers uses very little, if any, AI techniques, it solves a problem that human beings find very difficult to solve. Chemists usually learn to solve this problem by using visual intuition. The mathematics involved are deep enough so that the average chemist will not have the patience necessary to learn enough about the algorithm to use its insights in enumerating stereoisomers. One of central problems for AI work in chemistry now is how to use this new facility in structure elucidation.

EXAMINE

Often in the course of a structure elucidation problem, a large number of candidate structures, perhaps a hundred or more, are generated; and additional constraints must be derived, either from further data analysis or from new experiments. The EXAMINE function written by Neil Gray is used from within CONGEN to survey, classify, display, or discard structures. This function is very useful to the chemist who is searching for features common to a large number of the structures or for features that are unique to certain structures. The insights gained from using EXAMINE can be used in planning new experiments or in further data analysis. In pursuit of these objectives, the chemist can define functional groups and other structural features, or he can work with a predefined library of them. The EXAMINE function is then called, and it examines the list of candidate structures for the presence or absence of these features.

For example, the chemist can ask EXAMINE to look for all structures with exactly one labile proton. (A labile proton is a hydrogen atom attached to a nitrogen atom or a hydrogen atom attached to an oxygen atom.) The chemist can represent this structure in EXAMINE as an exclusive OR statement: exactly one hydrogen attached to an oxygen atom in the structure OR (exclusive) exactly one hydrogen attached to a nitrogen atom in the structure.

The user can then request EXAMINE to draw those structures that have this characteristic and those that do not, in order to produce summary statistics on its frequency of occurrence or to discard those structures with or without it. While CONGEN is always able to discard or prune away structures that do not satisfy certain constraints, EXAMINE provides the interactive ability to develop boolean combinations of constraints for pruning, substructure search, or subsequent classification.

REACT

Before spectroscopy became a major tool of the structural chemist, all structure elucidation had to be done by means of reaction chemistry, and it is still a major tool in solving structures. REACT is an interactive program written by Tomas Varkony, Dennis Smith, and Carl Djerassi (Varkony, Smith, and Djerassi, 1978). Although it is a close relative to the synthetic programs described below (see article C4), its purpose is to aid chemists in the structure elucidation task rather than to aid them in finding new synthetic routes.

To show how REACT can be used to reduce the number of candidate structures found by CONGEN, consider the following example. A dehydration reaction can be expressed as a production rule of the form: "If you see the pattern C-C-O, convert it to the pattern C=C." We now suppose that a dehydration reaction was applied to the unknown in question and yielded three distinct structures, which happened because the pattern C-C-O occurred in the molecule in three different places. This information can be used to eliminate structures from those under consideration: The structure list generated by CONGEN is passed to REACT; the dehydration reaction is defined by the user and then applied to all the candidate structures; those that do not yield exactly three products can be eliminated from consideration as candidate structures.

Although REACT does not contain stereochemical information, conformational information, or electronic information (the electro-negativities of its atoms and groups), it still can be used reliably in its structure elucidation function. Reactions used for structure determination tend to have high yield, to be reliable, and to involve simple separations. The reactions operate under a wide variety of conditions and usually involve rather simple changes to the unknown molecule. Thus, the perception routines do not need the sophisticated stereochemical, conformational, and electronic information of the organic synthesis programs discussed above.

Summary

Research in the DENDRAL project has followed two themes: To build a performance program for analysis of molecular structures, and to explore some problems of scientific inference using AI methods. The performance of Heuristic DENDRAL has been evaluated in the same way as that of a research chemist: by publications. (See the conclusion of the article C2a on DENDRAL for references.) In addition, CONGEN is used daily by chemists to aid in solving structure elucidation problems.

Because of the combinatoric size of analysis problems, exhaustive problem-solving methods were not an option, and much thought was given to the knowledge that enabled chemists to solve these problems. DENDRAL was one of the first programs to demonstrate

the power of encoding domain-specific, heuristic expertise, and was therefore one of the first projects to recognize knowledge acquisition as a major problem in AI (Buchanan, Sutherland, and Feigenbaum, 1969; Davis, 1976). The next article (C2c) discusses automatic inference of rules as one solution to the knowledge acquisition problem.

References

In addition to the DENDRAL references in the previous article, the following may be of interest: Brown, Masinter, and Hjelmeland (1974), Brown and Masinter (1974), Carhart et al. (1975), Carhart and Smith (1976), Masinter et al. (1974), Sheikh et al. (1970), and Smith and Carhart (1978).

C2c. Meta-DENDRAL

The domain-specific rules that constitute DENDRAL's knowledge about mass spectrometry were derived from consultation with experts in that field. Since the consultation process is time consuming, two alternatives to "handcrafting" knowledge bases were explored. One is interactive transfer of expertise (see article B). The other is automatic theory formation. Meta-DENDRAL is a program of the latter type. The rule formation task that Meta-DENDRAL performs is similar to the task of grammatical inference, sequence extrapolation, and concept formation (Hunt, 1975; Hedrick, 1974; Winston, 1970). Programs that perform these tasks can all be thought of as "induction" programs because they formulate general rules (or concepts, or patterns) from examples.

Meta-DENDRAL is designed to infer theories (rulesets) for the Heuristic DENDRAL program (see article C2a), which represents knowledge about mass-spectrometry as production rules. Automatic rule formation was chosen as a paradigm for Meta-DENDRAL for two general reasons. First, this design poses interesting epistemological questions, and, second, it is an arduous task to derive rules from human consultants, especially when the task-domain has only a small number of experts (as is the case in mass-spectrometry).

Representation of Knowledge about Mass spectrometry

In DENDRAL, knowledge about the fragmentation processes in a mass spectrometer is represented in the form of *production rules*. Each rule specifies a bond *fragmentation* in a particular context in a molecule. These rules are used by DENDRAL during its *test* phase to predict mass spectral data points, given a certain molecular structure. For example, one simple rule is:

(R1) N - C - C - C ----> N - C * C - C

Rules are interpreted for each molecule in the following way:

(1) Find all places in the molecule that match the subgraph expressed by the left-hand side of the rule.

(2) For each match, break the molecule at the bond marked with an asterisk in the right-hand side of the rule and save the fragment associated with the atoms to the left of the asterisk.

(3) Record the mass of all saved fragments.

Note that no migration of atoms between fragments is predicted by (R1).

The language of processes (right-hand sides of rules) is relatively simple: One or more bonds from the left-hand side may break and zero, one, or more, atoms may migrate between fragments. The interpretation of rule R1 in the above example is straightforward: If a molecule contains a nitrogen atom and three carbon atoms bonded as N-C-C-C, then it will fragment in the mass spectrometer between the middle two carbon atoms, and the N-C fragment will be recorded in the spectrometer as a peak at the point in the spectrum corresponding to the molecular weight of this fragment.

Formation of Mass Spectral Rules

The task of Meta-DENDRAL is to infer rules (like R1 above) from empirical data. Meta-DENDRAL is provided with descriptions of the structures of a related set of molecules, and with the set of peaks produced by the fragmentation of each molecule in the mass spectrometer. From these data it infers a small and fairly general set of mass-spectral rules to account for the fragmentations of the molecules and the corresponding spectral peaks.

Training instances. In order to learn rules, the Meta-DENDRAL program is presented with many examples of actual I/O pairs from the mass spectrometer. Each I/O pair represents a molecular graph structure, together with a single data point from the mass spectrum for that structure. The rules to be learned constitute a representation of the relevant fragmentations in the mass spectrometer. Typically, the program starts with a training set of six to ten related molecules and their associated spectra, each containing 50-150 data points--peaks marking the masses of recorded fragments (and the relative abundance of fragments at those masses).

In a large molecule, rule (R1) may apply more than once. For example, the spectrum of CH3-CH2-CH2-NH-CH2-CH2-CH2-CH3 will contain data points at masses 72 and 86 corresponding to the two fragments derived from the application of this rule:

CH3-CH2-CH2-NH-CH2

and

CH2-NH-CH2-CH2-CH2-CH3 .

For a number or reasons, data points are not associated uniquely with a single fragmentation and atom migration process (rule). For example, a single process may occur more than once in a molecule (as above), or more than one process may produce identical fragments, producing peaks at the same mass points in the spectra.

Spectral Data Points and Mass-spectral Processes: Statistical and Semantically Constrained Associations

Purely statistical learning programs (Jurs, 1974) find associations indicated by the data without judging the meaningfulness of these associations. This feature can be advantageous; at times an investigator's bias inhibits his seeing associations, or an investigator may be looking for all possible associations. But it is a disadvantage when the number of associations is so large that the meaningful ones, unmarked, get lost in the crowd.

In contrast to statistical approaches, Meta-DENDRAL utilizes a *semantic model* of the domain. This model has been included for two important reasons. First, it provides guidance for the rule formation program in a space of rules that is much too large to search exhaustively and in a domain of input data that is often ambiguous. Second, it provides a check for the meaningfulness of associations produced by the program, in a domain where the trivial or meaningless associations far outnumber the important ones.

Semantic model of the domain. The base-level, or zero-order, theory of mass spectrometry states that every subset of bonds within a molecule may break and that the resulting fragments, plus or minus migrating atoms, will all be recorded. This zero-order model of mass spectrometry is not specific enough to effectively constrain the rule search. Therefore, some general guidelines have been imposed on it, the so-called *half-order* theory.

The half-order theory asserts that bonds will break and atoms will migrate to produce data points. This theory orders the break-and-migrate process according to the following constraints:

Constraints on fragmentations:
- Double bonds and triple bonds do not break.
- No aromatic bonds break.
- Only fragments larger than 2 carbon atoms show up in the data.
- Two bonds to the same carbon atom cannot break together.
- No more than 3 bonds break in any one fragmentation.
- No more than 2 complete fragmentations occur in one process.
- At most 2 rings fragment in a multiple-step process.

Constraints on atom migration:
- At most 2 hydrogen atoms can migrate after a fragmentation.
- At most 1 H2O unit is lost after any fragmentation.
- At most 1 CO unit is lost after any fragmentation.

One of the most helpful features of this model is its *flexibility*: Any of the parameters can be easily changed by a chemist with other preconceptions; any of these assumptions can be removed and, as discussed in the following section, additional statements be substituted or added. This power to guide rule formation results in the program's discovering only rules within a well-known framework; on the other hand, it also results automatically in rules meaningful to the domain.

A chemist will often know more about the mass spectrometry of a class of molecules than is embodied in the half-order theory. It is important then to be able to augment the program's model by specifying class-specific knowledge to the program. This capability provides a way of forming new rules in the context of additional intuitions or biases about mass spectrometry. A chemist can thus see the "most interesting" rules (as defined by the augmentations) before the other rules. For example, one might be interested first in rules that mention at least one nitrogen atom before the numerous (and generally less interesting) rules that mention only carbon and hydrogen substructures.

Learning strategy. The Meta-DENDRAL program is based on a generator of production rules that uses predetermined syntax operating under the constraints of a semantic world model. The operation of Meta-DENDRAL can be summarized as follows:

Input

a. the structure of each of a set of related molecules (recall that Meta-DENDRAL is not a structure elucidation program but infers rules of mass spectrometry, which associate molecular structures and their mass spectra),

b. the spectral data points (peaks) for each of the molecules, and

c. the half-order theory (or some semantic theory to constrain the generation of rules).

Step 1. (INTSUM)

For each molecule, explain each peak in its spectrum by finding one or more fragmentation processes that would account for the peak. The number of plausible fragmentation processes is limited by:

a. considering only the fragmentations which are allowed by the half-order theory (e.g., no spectral peak can be explained by a fragmentation process that involves breaking a double bond), and

b. considering only fragmentations which produce fragments with a molecular weight corresponding to the weight represented by the peak. (Recall that each peak in a mass spectrum represents a number of molecular fragments of a given mass.) For example, if the total weight of the molecule under inspection is M, and the spectrum has a large peak associated with a molecular weight of M-47 mass units, then the only fragmentation processes considered as explanations for this point would be those that produce a fragment with a molecular weight of M-47. The tens, or hundreds, of other processes that fragmentations are consistent with the half-order theory, like cleaving off a hydrogen atom, are not even considered.

After each data point in the spectrum for each molecule has been explained by a plausible fragmentation process, the list of processes is summarized, since the same fragmentation processes will often be found to account for many spectral data points. The final product of INTSUM is a list of fragmentation processes with the total evidence for each such process.

Step 2. (RULEGEN)

The rules provided by INTSUM each account for a single fragmentation process in the context of a single molecule. As such, they are not general. The problem with general rules, on the other hand, is that a single one may subsume several of INTSUM's very specific fragmentations, *but also* fragmentations not represented in the set produced by INTSUM. That is, a general rule may correctly explain many data points in mass spectra, (positive evidence), but may also predict points that do not occur in any of the spectra (negative evidence). The purpose of RULEGEN is to find a set of rules which are more general than those of INTSUM, using positive evidence as a criterion of success. Negative evidence introduced by these rules is handled by a later step, called RULEMOD.

RULEGEN works by "growing" a tree of fragmentation rules, starting with one that is overly general and adding features to it so that it becomes more constrained. The rule that RULEGEN starts with is X * X, that is, the bond between any atoms will break, and the mass of fragment X will be recorded in the mass spectrometer as a peak. Obviously, every fragmentation rule is a specialization of this one, and it is too general to be interesting. But by specifying values for four features--the identity of X, the number of non-hydrogen neighbours X has, the number of hydrogen neighbors X has, and the number of doubly bonded neighbors X has--the general rule X * X can be "grown" into something more interesting.

Step 3. (RULEMOD)

RULEGEN can generate rules that predict nonexistent data points in the mass-spectral data. This negative evidence is the cost of the coarse method used by RULEGEN to find general rules. RULEMOD "tidies up" the rules produced by RULEGEN by merging rules, eliminating redundancies, and making rules more specific or general. In addition, if a rule has been used succesfully for a time, but an instance is found in which it is inappropriate, RULEMOD can modify the rule accordingly.

Output.

Output is a set of mass spectral fragmentation rules which are specialized enough to be interesting, but general enough to be efficient and nonredundant.

The Meta-DENDRAL program

The program itself is organized as a series of *plan-generate-test* steps, as found in many AI systems (Feigenbaum, Buchanan, and Lederberg, 1971). After pre-scanning a set of several hundred molecular structure/spectral data-point pairs, the program searches the space of fragmentation rules for plausible explanations and then modifies its rules on the basis of detailed testing. When rules generated from a training set are added to the model and another block of data is examined, the rule set is extended and modified further to explain the new data. The program iteratively modifies rules formed from the initial training set (adding to them); but it is currently unable to "undo" rules.

Integrating Subsequent Data. A requirement for any practical learning program is the ability to integrate newly acquired data in an evolving knowledge base. New data may dictate that additional rules be added to the knowledge base or that existing rules be modified or eliminated. New rules may be *added* to the rule base by running RULEGEN on the new data and then running RULEMOD on the combined set of new and previously generated rules.

When an existing rule is *modified*, it is important to maintain the integrity of the modified rule over past training instances. Consider the following example: A new training instance is acquired and, after credit assignment questions are resolved, it is decided that rule R was incorrectly "triggered" by some situation S. The left-hand side of rule R must be modified so that it will no longer match S. In general, there would be many changes possible to R that would kill the match to S, but some are better than others. The correct changes to R are those that do not alter past correct applications of R. Of course there is no way of knowing which of the possible changes to R will turn out to be correct for future data; and once a change is selected, the possibility still exists for backtracking at some future point.

A method has been developed for representing all versions of the left-hand side of a rule that are consistent with the observed data for all iterations thus far (Mitchell, 1977). This representation is referred to as the *version space* of the rule. By examining the version space of R, one can answer the question "Which of the recommended changes to R will preserve its performance on past instances?" The answer is simply "Any changes that yield a version of the rule contained in the version space." Using version spaces avoids the problem of selecting a single unretractable modification to R and therefore eliminates the

need for backtracking. For example, all the elements of the version space that match some negative instance S are eliminated. Similarly, when new data are encountered in which a situation S' is found to correctly trigger R, only those elements of the version space that match S' are retained.

Results

One measure of the proficiency of Meta-DENDRAL is the ability of a DENDRAL program using the learned rules to predict correct spectra of new molecules. One of the DENDRAL performance programs ranks a list of plausible hypotheses (candidate molecules) according to the similarity of their predictions (predicted spectra) to observed data. The rank of the correct hypothesis (i.e., the molecule actually associated with the observed spectrum) provides a quantitative measure of the "discriminatory power" of the rule set.

The Meta-DENDRAL program has successfully rediscovered known, published rules of mass spectrometry for two classes of molecules, including the aliphatic amines used as examples above. More importantly, it has discovered new rules for three closely related families of structures for which rules had not previously been reported. These are the mono-, di-, and tri-keto androstanes which share the common structural skeleton shown in Figure 1.

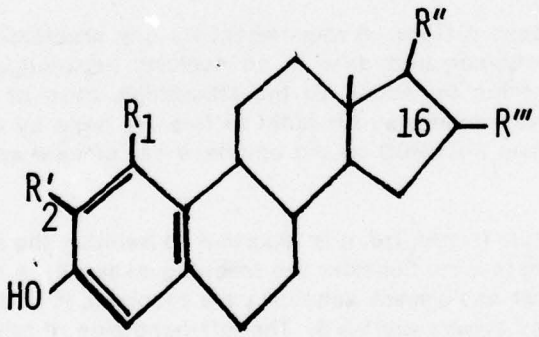


Figure 1. Structural Skeleton for Three Classes of Androstanes.

Meta-DENDRAL's rules for these classes have been published in the chemistry literature (Buchanan et al., 1976). Evaluations of all five sets of rules are discussed in that publication. This work demonstrates the utility of Meta-DENDRAL for rule formation in mass spectrometry for classes of structures.

The recent application of Meta-DENDRAL has been to a second spectroscopic technique: *13C-nuclear magnetic resonance* spectroscopy (Mitchell, 1978). This new version

provides the opportunity to direct the induction machinery of Meta-DENDRAL under a model of 13C-NMR spectroscopy. It generates rules that associate the resonance frequency of a carbon atom in a magnetic field with the local structural environment of the atom. Note that for 13C-NMR spectroscopy there is no requirement for a half-order theory since there is no equivalent to the fragmentation processes which occur in mass spectroscopy. Each data point is assigned to a unique atom in the molecule prior to the Meta-DENDRAL run. Thus there is no analog of the INTSUM phase which is required by the mass spectroscopy version. Instead, an assigned spectrum (atoms to data points) is given directly to RULEGEN.

13C-NMR rules have been generated and used in a candidate molecule-ranking program similar to the one described above. 13C-NMR rules formulated by the program for two classes of structures have been successfully used to identify the spectra of additional molecules (of the same classes, but outside the set of training data used in generating the rules). The rule-based molecule-ranking program performs at the level of a well-educated chemist in both the mass spectral and 13C-NMR domains.

References

See Lindsay, Buchanan, Feigenbaum, and Lederberg (forthcoming) for a thorough and current treatment of the DENDRAL programs. Buchanan and Feigenbaum (1978) is a recent, short description of the programs. The thesis by Mitchell (1978) is also recommended.

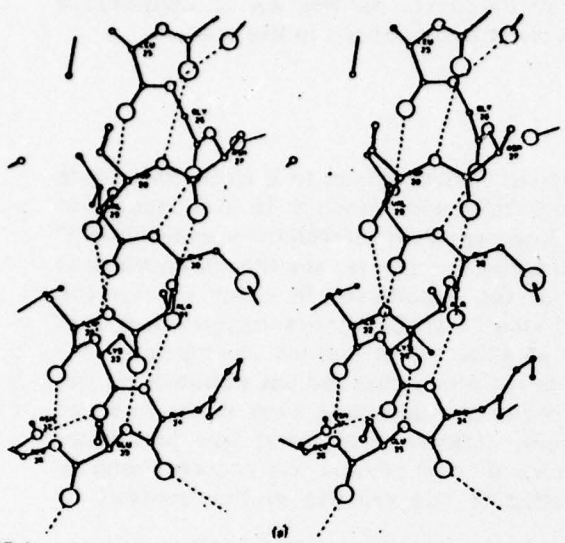
C3. CRYSALIS

The CRYSALIS system, which is still in the development stages, is an attempt to apply Artificial Intelligence methodology to the task domain of protein crystallography. Although the computer has been an essential tool in x-ray crystallography research for many years, nearly all its applications have been in the areas of data collection, data reduction, Fourier analysis, graphics, and other essentially numerical tasks (Feigenbaum, Engelmore, and Johnson, 1977). Those aspects of molecular structure inference that require symbolic reasoning or that use a significant amount of judgmental knowledge have traditionally been performed manually. A prime example is the task of *electron density map* interpretation.

In the course of deriving a protein structure, the crystallographer generates an electron density map, a three-dimensional description of the electron density distribution of a molecule. Due to the resolution imposed by the experimental conditions, the electron density map is an indistinct image of the structure that does not reveal the positions of individual atoms. The crystallographer must interpret the map in light of auxiliary data and general principles of protein chemistry in order to derive a complete description of the molecular structure. The goal of the CRYSALIS system is to integrate these diverse sources of knowledge and data to try and match the crystallographer's level of performance in electron density map interpretation. Automation of this task would shorten the time taken for protein structure determination by several weeks, to months, and would fill in a major gap in the construction of a fully automated system for protein crystallography.

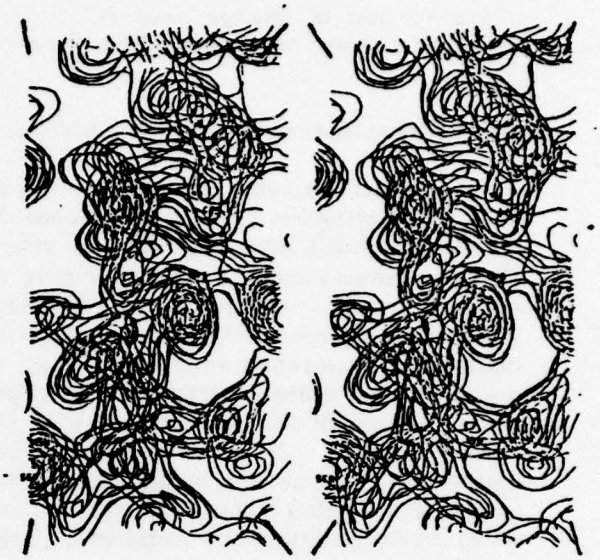
Description of the problem

When crystallographers use the term "electron density map," they usually have in mind some pictorial representation of the electron density defined over a certain region of space. The most commonly used representation is a three-dimensional contour map, constructed by stacking layers of conventional two-dimensional contour maps drawn on transparent sheets. By carefully studying the map, the experienced protein crystallographer can find features that allow him to infer approximate atomic locations, molecular boundaries, groups of atoms, the backbone of the polymer, etc. After several weeks (or months), he has built a model of the molecular structure that conforms to the electron density map and is also consistent with his knowledge of protein chemistry, stereochemical constraints, and other available chemical and physical data (e.g., the amino acid sequence). Figure 1 shows a portion of a protein structure and the associated electron density map from which it was inferred.



(a)

FIG. 1 A stereo-view of the electron density (b) at 2.8 Å of an α-helix in lysozyme, and the molecular structure (a) corresponding to this density (Snape, 1974).



(b)

FIG. 1 (*continued*)

The automation of this task would require a computational system that could generate its own structural hypotheses, as well as display and verify them. This capability requires: (a) a representation of the electron density function suitable to machine interpretation, (b) a substantial chemical and stereochemical knowledge base, (c) a wide assortment of model-building algorithms and heuristics, (d) a collection of rules and associated procedures for using this knowledge to make inferences from the experimental data, and (e) a problem-solving strategy for applying the knowledge sources (KSs) effectively, so that the appropriate procedures are executed at the times that they are most productive.

Protein crystallographers who build models move continually across a large field of basic facts, special features of the data and implications of the partial model(s) already built, looking for any and all opportunities to add another piece to their structure. There are several desiderata to working in this "opportunistic" mode of hypothesis formation: (a) The inference-generating rules and the strategies for their deployment should be separate,

(b) the rules should be separate from the mechanics of the program in which they are embedded, and (c) the representation of the hypothesis space should be compatible with the kinds of hypothesis-generating rules available. The modularity of such a system would allow users to add or change rules for manipulating the database, as well as to investigate different solution strategies without having to make major modifications to the system.

The CRYSALIS Architecture: The Blackboard

A problem-solving paradigm that meets the above specifications, to a large degree, is that of HEARSAY-II (see article Speech.D2)--specifically with respect to the issues of knowledge integration and focus of attention. In Hearsay-II, an "iterative guess-building" process takes place: A number of different *knowledge sources* (facts, algorithms, heuristics) cooperate when working on various descriptions of the hypothesis. In order to use the knowledge sources efficiently, a global database--the "blackboard"--is constructed that contains the currently active hypothesis elements at all levels of current description. The decision to activate a particular knowledge source is not preestablished but depends on the current state of the solution and what available knowledge source is most likely to make further progress. The control is, to a large extent, determined by what has just been learned: A small change in the state of the "blackboard" may provide the preconditions to instantiate further knowledge sources (an illustration of this process in the context of electron density map interpretation is given below).

Figure 2 shows the types of data and hypotheses that are used in CRYSALIS. As in Hearsay-II, the hypotheses are represented in a hierarchical data structure. In our case the different information *levels can* be partitioned into three, distinctly different "panels," but the concept of a globally accessible space of hypotheses is essentially the same for both systems. Figure 2 also illustrates how knowledge sources (only a small subset is shown) play the same role as in Hearsay-II: adding, changing, or testing hypothesis elements on the blackboard. Further explanation of these diagrams is given in Engelmore and Nii, 1977. The processes of generating or modifying hypotheses and of invoking knowledge sources is nearly identical to those described for the AGE system (Nii and Aiello, 1978).

Representation of Knowledge in the System

As mentioned above, there are many diverse sources of information used in protein structure inference. The problem of representing all the knowledge in a form that allows its cooperative and efficient use in the search for plausible hypotheses is of central concern to the developers of CRYSALIS. The system currently under development draws upon many concepts that have emerged in the design of other large knowledge-based systems--e.g., the use of production rules and blackboards. We describe here how these concepts have been adapted to our particular task.

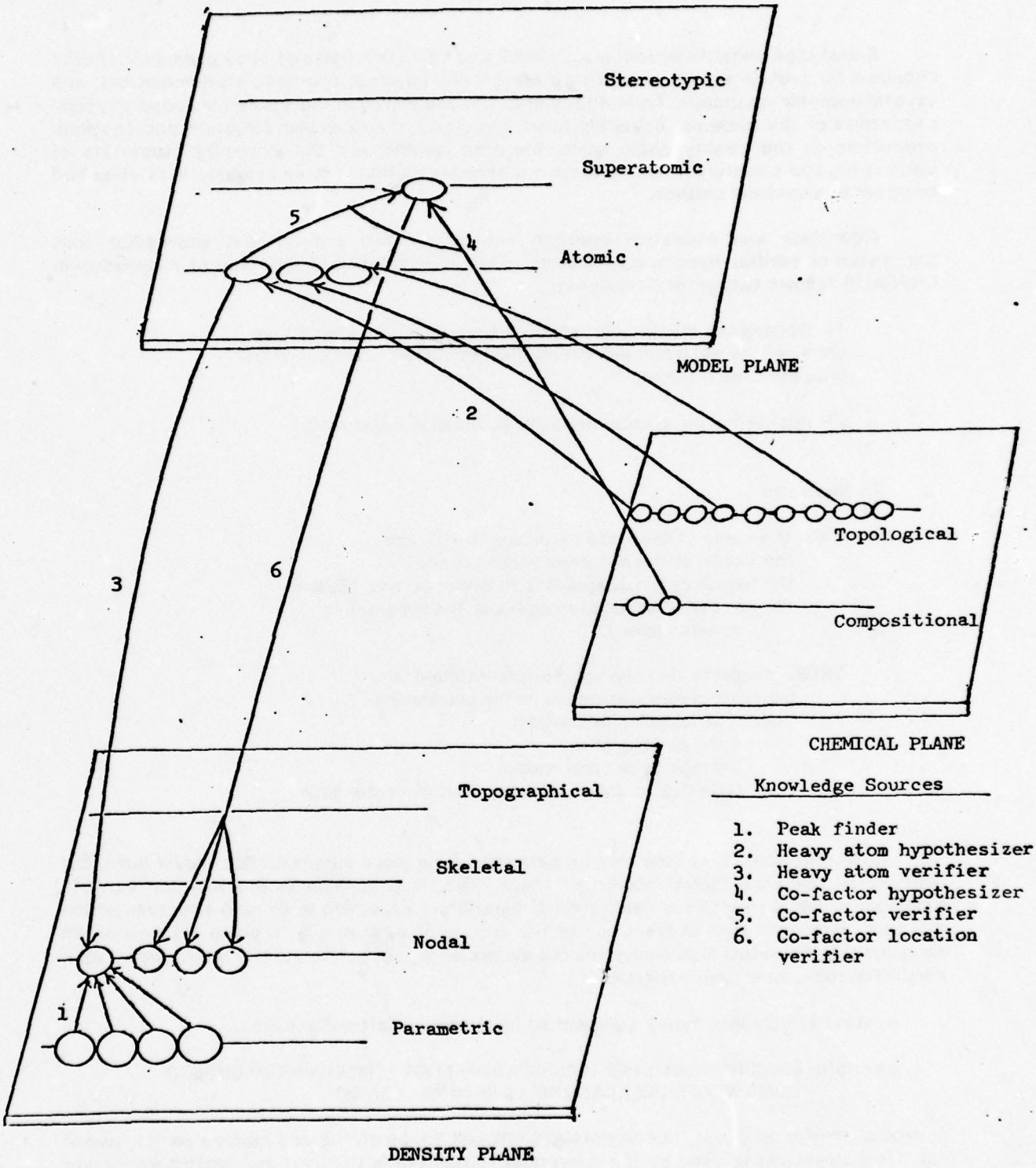


Figure 2. Panels of the CRYSALIS blackboard, and examples of the application of knowledge sources.

Knowledge consists of facts, algorithms, and heuristics (rules of good guessing). Facts required for protein structure inference are general physical, chemical, stereochemical, and crystallographic *constraints*. Typical factual knowledge stored in the system includes physical properties of the elements commonly found in proteins, the molecular structure and chemical properties of the twenty amino acids, the bond lengths, and the symmetry properties of various crystal structures. These facts are encoded as tables or as property lists attached to specific structural entities.

Algorithms and heuristics comprise both the formal and informal knowledge that generates or verifies hypothesis elements. The representation of this type of knowledge in CRYSALIS follows two general principles:

1) Decompose identifiable areas of knowledge into elementary units, where each unit increments the hypothesis when specified preconditions are met.

2) Represent the elementary units as *situation-action rules*.

To illustrate:

IF: the name of the current-residue is GLU, and
the shape of the subgraph is forked, and
the length of the subgraph is between 40 and 75, and
the number of associated peaks of the subgraph is
greater than 1

THEN: conclude that the subgraph is matched, and
generate a new superatom on the blackboard,
with the following properties:
Type is 'side-chain
Belongs to current-residue
Data-link to subgraph with certainty factor 500

Note that several actions may be performed for a given situation. Not shown here, but present in the LISP implementation of these rules, is a position in the rule for variable bindings, to avoid repetitious calculation of parameters appearing in several situation-action clauses. Also note that at least one of the actions of each rule is to place a token on an *event list*. In the actual implementation, the syntax of the "action" clause is represented as a single function. An example follows:

syntax: (<inference type> <element being changed> <att-value pairs>)

example: (SUBGRAPH.MATCHED (GENSUPATOM)((TYPE 'SIDECHAIN)(BELONGSTO CURRENT.RESIDUE)(DATALINK (SUBGRAPH . 500))))

In this example, an event, SUBGRAPH.MATCHED, will be generated and queued on the event list. The event-list is used by the interpreter (discussed in the next section) to determine what to do next, that is, which set of knowledge sources to invoke after the current event has been processed.

Event-driven control

The CRYSALIS system uses an *event-driven control structure*. In this scheme, the current state of the hypothesis space determines what to do next. The monitor continually refers to a list of current events--the event-list--that is used to trigger those knowledge sources most likely to make further headway. As a knowledge source makes a change in the current hypothesis, it also places an item on the event-list to signify the type of change made. Thus, as events are drawn from the event-list for processing, new events are added, so that under normal conditions the monitor always has a means for choosing its next move.

The normal iterative cycle of problem solving uses the event-list to trigger knowledge sources, which create or change hypothesis elements and place new events on the event-lists. The system's behavior is *opportunistic*: it is guided primarily by what has been most recently discovered, rather than by the requirement to satisfy subgoals. An event-driven control structure was chosen partly to be efficient in selecting appropriate knowledge sources, and partly to conform with the structure-modeling process normally employed by protein crystallographers.

Rules

The formal and informal procedures that comprise our knowledge sources are expressed as rules, as discussed above. These rules are collected into sets, each set being judged appropriate to use when particular types of events occur. The events generally reflect the level at which the inference is being made, which in turn reflects the model's level of detail. The correspondence between *event classes* and rule sets is established by another set of rules, the *task rules*. The task rules are used to decide which KS or sequence of KSs to call in order to perform one of the typical tasks in building the structure--e.g., tracing the protein backbone between two anchor points. The decision is based on the state of the blackboard and the items on the event list. The task rules thus form a second layer of rules, which directs the system's choice of knowledge sources for a given event, reflecting the system's knowledge of what it knows.

Once a task is either completed or fails, the system looks to a higher level of control to determine what to do next. At this higher level--the strategy level--the structure-building process can either try to solve the current subproblem by another method or shift attention to another region of the structure. Strategy level decisions are also expressed as rules and make use of the current state of the blackboard and event list. For example, one strategy rule is:

> IF: the initialization task is complete, and
> the locations of two or more atoms are known
> (also called 'toeholds'), and
> these toeholds are separated by less than 6 residues
> in the amino acid sequence, and
> none of the intervening residues are identified from the data,
>
> THEN: select the two-point chain-tracing task and focus on the
> subsequence bounded by the toeholds.

The part of the monitor that interprets and obeys the event rules may be likened to a middle-level project manager who knows which specialists to call in as new, partial solutions to a particular problem are discovered. Continuing the analogy, the middle-level manager occasionally gets stuck and needs help from higher level management. As mentioned earlier, some high-level decision (such as merging two or more events to produce a new event or shifting attention to another part of the blackboard), is required. This level of decision making is embodied in a set of strategy rules, which are used to direct the top-level flow of control. We thus have a completely rule-based control structure that employs three distinct levels of rules (or knowledge): the specialists, commonly called the knowledge sources; the task rules, representing knowledge about the capabilities of the specialists; and the strategy rules, which know when to use all available knowledge to solve the problem. Although this pyramidal structure of rules and meta-rules could continue indefinitely, the flexibility of knowledge deployment offered by our three-tiered system appears sufficient for this problem-solving system. Similar ideas in a simpler context have been explored by Davis, 1976 for the MYCIN system.

System Performance -- An Example

To give some indication of the system's current level of performance, we present an annotated typescript in which a typical hypothesis formation task is completed. The example is the subproblem of extending the model from an "island of certainty," or anchor point, by using the crytallographic data to determine where to extend the model in space and by using the amino acid sequence to generate expectations of features that ought to be present in that region. The knowledge sources invoked in this example use an abstraction of the density map called a *subgraph*. A subgraph is a collection of segments obtained from a skeletonized density map, which hopefully matches an identifiable substructure in the protein--e.g., a side chain. The amino acid sequence assumed here is METhionine, LYSine, LYSine, TYRosine, etc. (the example uses data from the protein Rubredoxin). The example starts after passing control to a knowledge source called ANCHOR.TOEHOLD. The toehold of interest in this case is the sulphur atom in the methionine sidechain. This toehold is just a point in space and must be connected to the skeleton.

```
INFERENCE: EVENT-1   BY RULE 1      IN RULESET ANCHOR.TOEHOLD

  EVENT NAME: TOEHOLD.ANCHORED
  CURRENT HYPOTHESIS ELEMENT: SA2
  NEW PROPERTIES:  ((TYPE SIDECHAIN) (BELONGSTO (MET . 1))
    (SEGS (((1 SEG240) . 100) ((1 SEG238) . 100))) (MEMBERS (A3)))
```

The ANCHOR.TOEHOLD knowledge source has found subgraphs of the skeleton, but its limited knowledge cannot assign much certainty to the inference. The "real" matching of skeleton parts with expected residue is accomplished by MATCH.SDCHN. This knowledge source uses the shape of the subgraph, its length, the number of peaks associated with the candidate subgraph, and their heights. If a certainty factor (CF) of 500 or more is assigned, the sidechain is considered located (CF's have a range of -1000 to 1000; the CF combining function being the same as that used by MYCIN; see article Medicine.C2).

```
INFERENCE: EVENT-2   BY RULE 3  IN RULESET  MATCH.SDCHN
```

EVENT NAME: TOEHOLD
CURRENT HYPOTHESIS ELEMENT: SA3
NEW PROPERTIES: (SEGS (((1 SEG238) . 823) ((1 SEG240) . 555))))

If a sidechain is found, the trace tries to find the alpha carbon location by finding a peak of a certain type near the root of the sidechain. The KS used to propose an alpha carbon position is called POSSIBLE.CALPHA. The system assumes that the location of this peak is a more accurate guide than the skeleton for locating this class of atom.

INFERENCE: EVENT-3 BY RULE 5 IN RULESET POSSIBLE.CALPHA

EVENT NAME: C.ALPHA
CURRENT HYPOTHESIS ELEMENT: A4
NEW PROPERTIES: ((TYPE C) (NAME CA) (BELONGSTO (MET . 1))
(D.PEAKS ((PK076 . 500))))

Once the toehold has been anchored, this trace becomes essentially a generate-and-test search, heavily constrained by the sequence. The basic control cycle for the trace is: Propose a sidechain, match it; propose a peptide, match that; and loop until a match fails. Sometimes the carbonyl group present in each peptide will appear as a small sidechain. If this happens, the proposed peptide will extend only from the last sidechain up to this pseudo-sidechain, and the peptide will fail to match. This failure prompts the system to try matching the "sidechain" as a carbonyl. Success of this match would mean that only half of the peptide has been found; the system can then propose a larger peptide, which contains the old one, and proceed as before.

INFERENCE: EVENT-4 BY RULE 4 IN RULESET MATCH.PEPTIDE

EVENT NAME: PEPTIDE
CURRENT HYPOTHESIS ELEMENT: SA4
NEW PROPERTIES: ((TYPE PEPTIDE) (BELONGSTO (MET . 1))
(SEGS (((SEG6 SEG8) . 84))) (PEAKS (PK076 PK078)))

INFERENCE: EVENT-5 BY RULE 5 IN RULESET MATCH.CARBONYL.SC

EVENT NAME: CARBONYL.FOUND
CURRENT HYPOTHESIS ELEMENT: A5
NEW PROPERTIES: ((TYPE CO) (NAME CARBONYL) (BELONGSTO (MET . 1))
(SEGS (((1 SEG5) . 581))) (PEAKS (PK036)))

INFERENCE: EVENT-6 BY RULE 4 IN RULESET MATCH.PEPTIDE

EVENT NAME: PEPTIDE
CURRENT HYPOTHESIS ELEMENT: SA4
NEW PROPERTIES: ((SEGS (((SEG6 SEG8 SEG10) . 420)))
(PEAKS (PK076 PK078 PK036)))

INFERENCE: EVENT-7 BY RULE 7 IN RULESET MATCH.SDCHN

```
  EVENT NAME: SIDECHAIN
  CURRENT HYPOTHESIS ELEMENT: SA6
  NEW PROPERTIES: ((TYPE SIDECHAIN) (BELONGSTO (LYS . 2))
    (SEGS (((1 SEG242) . 527))))

INFERENCE: EVENT-8   BY RULE 5 IN RULESET POSSIBLE.CALPHA

  EVENT NAME: C.ALPHA
  CURRENT HYPOTHESIS ELEMENT: A6
  NEW PROPERTIES: ((TYPE C) (NAME CA) (BELONGSTO (LYS . 2))
    (D.PEAKS ((PK078 . 500))))

INFERENCE: EVENT-9   BY RULE 4 IN RULESET MATCH.PEPTIDE

  EVENT NAME: PEPTIDE
  CURRENT HYPOTHESIS ELEMENT: SA6
  NEW PROPERTIES: ((TYPE PEPTIDE) (BELONGSTO (LYS . 2))
    (SEGS (((SEG232 SEG16) . 600))) (PEAKS (PK017 PK125)))
```

(Three more events, similar to those preceding, have been omitted.)

```
INFERENCE: EVENT-13   BY RULE 6 IN RULESET MATCH.SDCHN

  EVENT NAME: SIDECHAIN
  CURRENT HYPOTHESIS ELEMENT: SA9
  NEW PROPERTIES: ((TYPE SIDECHAIN) (BELONGSTO (TYR . 4))
    (SEGS (((6 SEG212 SEG40 SEG36 SEG35 SEG228) . 502))))
```

The matching cycle terminates in one of two ways. If the skeleton becomes so overconnected that the access function cannot propose the next subgraph (sidechain or peptide), the trace fails; or if the certainty of a match is too low and there are no rules to save the situation, the trace fails. Upon termination, one final knowledge source is called to link together hypothesis elements belonging to the same residue, creating an organizing "backbone."

```
INFERENCE: EVENT-14   BY RULE 3     IN RULESET TRACE.CLEANUP

  EVENT NAME: LINK-CA-TO-PEPTIDE
  CURRENT HYPOTHESIS ELEMENT: SA4
  NEW PROPERTIES: ((MEMBERS (A4)))
```

(Two more events, like the preceding one are omitted here.)

```
INFERENCE: EVENT-17   BY RULE 7     IN RULESET TRACE.CLEANUP

  EVENT NAME: BACKBONE
  CURRENT HYPOTHESIS ELEMENT: ST1
  NEW PROPERTIES: ((TYPE BACKBONE) (CF 511) (DIRECTION 1)
    (RANGE (1 . 4)) (MEMBERS (SA1 SA2 SA3 SA4 SA5 SA6 SA7 NIL))
```

Summary

At the present time, CRYSALIS is capable of performing only a small portion of the total task of electron density map interpretation. The development and implementation of all the knowledge sources required for the complete task is a long-term effort. CRYSALIS currently contains a relatively small knowledge base that permits the interpretation of portions of high-quality, high-resolution (2.0 Angstroms or better) electron density maps. The system is expected to evolve toward an extensive knowledge-based problem solver capable of complete interpretation of medium-quality, medium-resolution (2 to 2.5 Ang.) electron density maps. Although CRYSALIS is not yet worthy of serious attention by the protein-crystallographic community, its defects lie primarily in its relatively meager knowledge base and not in its design. As new knowledge sources are added to the system, its level of performance is expected to rise to the point where its use will be a significant aid in the determination of new protein structures.

References

See Engelmore and Nii (1977), Engelmore and Terry (1978), and Feigenbaum, Engelmore, and Johnson (1977).

C4. Organic Synthesis

The synthesis of organic compounds is central to the creation of new chemical products and more efficient processes for manufacturing old products. However, the synthesis process for a particular product is typically very expensive to run and very hard to design. Therefore, there is great interest among both academic and industrial chemists in new tools to aid in finding new synthetic routes.

A synthesis problem begins with the structural description of a compound that someone wants synthesized, often because the compound has useful properties (e.g., a drug or a vitamin). Synthesis can also be a definitive confirmation of a postulated structure for an unknown compound in an analysis problem, since the synthesized compound and the unknown compound will, if identical, produce identical test results.

Chemists use the computer and AI techniques to systematically explore the *synthesis tree* and to help organize the immense body of available knowledge about chemical reactions. This approach of exhaustively exploring the interesting branches of the synthesis tree was called the logic-centered approach by Corey and Wipke, who first explored computer-aided organic synthesis. "Interesting" branches are those most likely to produce the desired result. "Interesting" is an extremely difficult concept to define and to cast into an algorithm, therefore, for now, the search must be guided interactively by the chemist. Some of the relevant considerations are: the efficiency of a reaction, the cost of materials, and the difficulty of meeting the experimental conditions that support a reaction.

The chemist represents the "target" structure graphically and relates it to simpler chemicals via known chemical reactions. He relates those to still simpler ones, until he reaches a set of commands, comparable to starting materials readily available from chemical supply houses or which can be easily synthesized in a few steps in the laboratory. One plan for synthesizing the compound, called a "synthetic route," may involve dozens of separate reactions. If the molecule is at all complicated there are an immense number of distinct synthetic routes. For example, a simple steroid composed of about 20 atoms has over 2.4×10^{18} possible direct routes .

Synthetic routes can be visualized using an AND/OR tree (see section Search.A2). The tree descends from the goal node, the target molecule, to the terminal nodes, equivalent to the starting materials. The branches connecting the nodes are chemical reactions. Since a synthesis plan involves combining compounds in reactions, the AND-links of the tree are present in any one synthesis route; alternative ways of making a compound anywhere within the plan are represented by OR-nodes.

The Three Major Programs

There are three major programs in computer-aided organic synthesis. The earliest is LHASA (Logic and Heuristics Applied to Synthetic Analysis), which was written by Corey and Wipke at Harvard and is maintained at Harvard by Corey and his research group. SECS (Simulation and Evaluation of Chemical Synthesis) is an outgrowth of LHASA, written by Wipke and maintained by Wipke and his research group at the University of California at Santa Cruz. It extended the LHASA paradigm by the inclusion of stereochemical and conformational information into all aspects of the computer program. The third major program is SYNCHEM

(Synthetic Chemistry), written and maintained by H. L. Gelernter and his research group at the State University of New York at Stony Brook. The main features of these three programs are summarized in Table 1.

Table 1
Chemical Synthesis Programs

Program	Principal Designer	Main Features
LHASA	E. J. Corey	Large procedural knowledge base of "transforms." Interactive, high-performance.
SECS	W. T. Wipke	Separate knowledge base of many "transforms" with special interactive language for defining new ones (ALCHEM). Interactive graphics, and high-performance.
SYNCHEM	H. Gelernter	Motivated by AI search problems. Evaluation during search done by the program, not by a chemist.

Since SECS was designed to extend the methods in LHASA, much of the discussion of SECS is true of LHASA. However, SECS has additional features that are of interest to computer scientists. Of the three, only SECS is demonstrably machine independent.

Two Different Approaches

A major distinction between SECS (and LHASA) and SYNCHEM is that the former is oriented to high performance, while SYNCHEM is oriented more to AI issues. As a consequence of this fact and the fact that chemists' intuitions about "interesting" pathways are hard to define, SECS relies on a chemist's interacting with the program. SYNCHEM, on the other hand, searches the space without interactive guidance from a chemist. (This is not to say that SECS and LHASA lack interest or that SYNCHEM is incapable of high performance.)

In operational terms, the main difference is whether the evaluation function for the search procedure is explicitly given to the program and used without guidance from the chemist (SYNCHEM) or whether the evaluation function is not explicitly given to the program (SECS and LHASA). These are called the noninteractive and interactive approaches below. SECS can be reconfigured to run noninteractively, although a chemist's guidance tends to give better results.

The Chemical Knowledge Base

The primary item of knowledge in chemical synthesis is the chemical reaction--a rule describing a situation in which a change can occur (to a molecular structure) plus a description of that change. For example, the reaction shown in Figure 1 describes a change to a molecule containing the substructure O=C-C-C=O in the presence of the reagent oxalyl chloride.

O=C-C-C=O + Oxalyl Chloride -----> O=C-C=C-CL

Figure 1. Graphical representation of a chemical reaction.

To design a synthesis route from starting materials to target molecule, knowledge of reactions can be used in either of two ways:

1. **Forward direction:** Apply known reactions to starting materials, then to the products of those reactions, the products of products, etc., until the target is reached. The combinatorics of this approach make it impossible in practice because there are thousands of possible starting compounds and only one target.

2. **Reverse direction:** Starting with the target molecule, determine which reactions might produce it. Then look for ways to make the precursors, and the precursors of precursors, etc., until starting materials are reached. Storing the reactions in the reverse direction makes it easier to search the tree of possible pathways.

All three programs have a large knowledge base of *reverse chemical reactions* called transforms--production rules of the condition-action form, with the left-hand side being a substructure pattern to be matched in the target structure (or intermediate structure) and the right-hand side being a description of precursors that will produce the goal structure under specified reaction conditions. Each of the three projects have dealt with the problems of constructing a knowledge base in very different ways.

1. The LHASA knowledge base is a set of procedures. Although it contains very sophisticated chemistry knowledge, it is difficult to modify.

2. The SECS knowledge base contains about 400 separate transforms. New transforms can be defined by users and entered into the knowledge base without changes to the program. Because of its clarity, it is used for illustration and is discussed in detail below.

3. The SYNCHEM knowledge base is a library of reactions that can be updated by chemists without reprogramming. Each reaction is automatically compiled into a reverse reaction. In addition, the knowledge base contains a large library of starting compounds that are available commercially.

Each of the SECS transforms is stored on external storage independent of the SECS program; this feature enables the knowledge base to be tailored to a specific problem domain. Further, the number and complexity of transforms is not limited by the size of core

memory. A simple, flexible language, called ALCHEM, is provided in which chemists can enter new transforms into the knowledge base.

ALCHEM embodies a model of what information is needed in order to adequately describe a reaction. According to this model, a transform consists of the following six sections:

(1) Transform name.

(2) Substructure key or pattern to be matched.

(3) Character--used to help judge the relevance to strategic planning.

(4) Scope and limitations.

(5) Reaction conditions--which must not be violated by the remainder of the molecule containing the substructure key.

(6) Manipulation statements--describing the graph transformations to be performed.

This will be clarified below with an example.

In the reaction shown in Figure 1, one of the Oxygens double-bonded to Carbon is replaced by a single bond to a Chlorine. To go from a graphical representation of a synthetic reaction to the graphical representation of a SECS transform, we reverse the left- and right-hand sides and specify additional important conditions. Using the ALCHEM language, the Chemist could interactively enter the following representation of this transform.

```
Comment: Chloroenones, O=C-C=C-CL goes to O=C-C=C-CL
Reagent: Oxalyl Chloride
Ref: Heathcock and Clark (1976).

Transform name:        CHLOR-ENONE
Substructure key:      O=C-C=C-CL  <1 = 2 - 3 = 4 - 5>
Priority:          100

Character:             CHARACTER ALTERS GROUP

Scope                  IF ACID IS OFF PATH THEN KILL
and Limitations:        IF ESTER IS OFF PATH THEN KILL
                   IF HYDROGEN IS ALPHA TO ATOM 4 THEN
                        BEGIN
                        IF HYDROGEN IS ALPHA TO ATOM2
                        THEN SUBTRACT 75 FROM PRIORITY
                        DONE

Manipulation:          BREAK BOND 3
Statements:             DELETE ATOM 5
                   ADD O OF ORDER 2 to ATOM 4
```

In the actual reaction, of course, the chlorinated compound comes from the precursor.

Referring to the manipulation statements, "BREAK BOND 3" refers to the third bond from the left in the substructure key; the double bond between the two carbons is reduced to a single bond. Similarly, "DELETE ATOM 5" refers to the Chlorine atom CL, the fifth atom from the left. When the program is actually run, a compiler called SYNCOM translates the ALCHEM statements into machine readable form before the program is run.

A Brief Description of SECS

SECS and LHASA have been designed to divide the work between the chemist and the computer in the most optimal way. In a recent paper Wipke et al. (1977), explain their philosophy.

> Our performance goal for the program was that the program should be able to help a chemist find many more good and innovative syntheses than the chemist could working alone. Because of the complexity of the problem domain, we felt the chemist and computer working together with each assigned tasks for which they are best suited, and with efficient interaction between the two, would be more effective than either working alone. Our goal was not to replace the chemist, but to augment the chemist's problem solving capabilities.

Graphics. The chemist communicates with the SECS program using a graphics terminal with a CRT, a mini-computer, a keyboard, and a light pen. Using the pen, the chemist draws on the screen the graphical structure of the "target" molecule to be synthesized. Much effort has gone into human engineering. The SECS graphics routines are designed to be as near as possible to the chemists' normal modes of thought, which is the structure diagram or the molecular model. There are similar facilities in LHASA. By convention, hydrogen atoms are suppressed, as discussed above. Another convention is that only noncarbon atoms (called "heteroatoms") are labeled. This convention is useful since the majority of nonhydrogen atoms in organic molecules are carbon.

Application of a Transform. Applying a transform is not simply a matter of matching the substructure key to a molecule and, if the subgraph fits, executing the graph manipulation statements. The scope and limitations determine much of the context in which the transform will be applicable. Also, it is necessary to check three-dimensional information and electronic environment information (that is, the tendency of the atoms in the molecule to be positively or negatively charged) in order to make an accurate assessment about whether a transform is applicable. A common situation in synthetic chemistry is that we have a functional group to modify and a reagent to change it, but the functional group is hindered (spatially) by another functional group or another portion of the molecule. In such cases, the reagent molecules cannot react with the group and change it; although they might in other spatial contexts.

Without the three-dimensional information given by the so-called "model-building" routines, the program has no way of knowing that the transform cannot apply. After the spatial modeling has been done, the program can perceive that even though the required

functional group is present, the transform cannot be applied directly because it is inaccessible to the reagent molecules. If the transform is very high priority, a means-end analysis can be done to find ways of altering the molecule, so that the given functional group is accessible.

A Brief Description of SYNCHEM

The aims of Gelernter's group on SYNCHEM are stated very clearly in Gelernter et al., 1977:

> Extraordinarily rapid progress during the early stages of an attack on a new problem area is a rather common occurrence in AI research; it merely signifies that the test cases with which the system has been challenged are below the level of difficulty where combinatorial explosion of the number of pathways in the problem space sets in....It is the goal of AI research to move that threshold higher and higher on the scale of problem complexity through the introduction of heuristics--heuristics to reduce the rate of growth of the solution tree, heuristics to guide the development of the tree so that it will be rich in pathways leading to satisfactory problem solutions, and heuristics to direct the search to the "best" of these pathways.

SYNCHEM is noninteractive. The molecule to be synthesized is input, and the program uses heuristic search to look for the best synthetic route. The program decides which node of the tree to develop further, by estimating the "cost" of reaching the goal from that node plus the estimated "cost" of reaching that node from starting materials. One of the interesting AI issues is that the program's definition of "cost" depends on the context of the problem as well as on static features such as efficiency of reactions, the monetary cost of materials, etc. For example, costs are measured differently in an exploratory research context than in an industrial production context.

The long-range hope of the SYNCHEM group is that the study of AI in this domain will lead to new insights in AI and also eventually to a noninteractive system that will be of use to chemists.

SYNCHEM Solution Evaluation. The following quotation (Gelernter et al., 1977) illustrates the difference between organic synthesis and a more familiar domain such as theorem proving.

> Unlike much of the earlier work in problem-solving....where any valid sequence of transformations from premises to goal provided an acceptable solution, we were not to be satisfied by an indicated synthesis route of very low yield, or one requiring difficult or inefficient separations of goal molecules from by-products along the way, at least not before the machine had tried and failed to find a more efficient procedure of higher yield....It is the question of relative merit of proposed solutions under the constraints of the problem that represents a substantial departure from most of the work reported in the literature of artificial intelligence.

The complexities of the domain are highlighted by the fate of one of the most significant results produced by the program. SYNCHEM proposed a synthetic route for a naturally occurring antibiotic that was at that time under development by A. R. Rinehart's group at the University of Illinois. The route was considered interesting enough to merit a laboratory investigation. However, the laboratory attempt failed. One of the crucial steps in the synthesis route could not be accomplished in the laboratory and the proposed route had to be reluctantly abandoned. No successful routes to the molecule have yet been found. All synthetic routes, whether proposed by a computer program like SYNCHEM or by a person, are provisional until they can be verified by experiment.

SYNCHEM Search Strategy

SYNCHEM'S search strategy algorithm first expands the goal node to find all its precursors. Next it computes the cost of reaching the target molecule from the precursors, taking into account the efficiency and difficulty of the reactions. It also estimates the difficulty of synthesizing the precursor nodes from the available starting materials. Subgoal selection criteria are a function of both the accumulated heuristic estimates of reaction merit and yield along the path from subgoal to goal, and of a prediction of the probable reaction merit and yield along the best path from starting materials to the subgoal. SYNCHEM updates the merit ratings with information associated with each intermediate structure. Merit, as mentioned above, is based on most recent estimates of compound complexity (i.e., difficulty in synthesizing it) and reaction path merit (yield, cost, etc.) after each cycle of subgoal generation. The selection of a new subgoal always begins with a new scan of the tree from the top. Thus the search is performed in a best-first manner: If newly acquired information changes the ratings for subgoals, the next subgoal selected can lie on a completely different branch of the tree. In this way, the program will never develop an unfortunate choice (pathway down to starting materials) before backtracking and exploring more fruitful branches.

Summary

Computer-aided chemical synthesis is a potentially powerful new tool for both research and industrial chemists. The utility of any of the programs discussed here critically depends on the size and accuracy of their knowledge base of organic chemical reactions. Although far from complete, the knowledge bases now contain highly detailed descriptions of numerous synthetic reactions. All of the programs have convincingly demonstrated their ability to find plausible synthetic routes for important organic materials, often in less time than chemists working alone. The SECS program has a user community of chemists in Europe and North American, who add new transforms as well as use the program for synthesis planning. The effort spent on human engineering for chemists has made it possible for chemists to use the program effectively (and want to use it) and independently of the program's designers. One of the long-range hopes of chemists and computer scientists working in computer-aided organic synthesis is that this work on knowledge bases will lead to an improved classification of chemical reactions.

Because the heuristic search paradigm fits the synthesis planning problem well, AI research has had much to offer. In addition, current AI work on knowledge-based expert systems provides concepts and tools for representation and management of those large, ever changing sets of chemical facts and relations.

References

See Cory and Wipke (1969), Gelernter et al. (1977), Gund, Andose, and Rhodes (1977), and Wipke et al. (1977).

D. Applications in Mathematics

D1. AM

AM is a computer program written by Douglas Lenat (1976) that explores the field of elementary mathematics, enlarging its vocabulary of objects and operators by defining new ones, gathering empirical data about the concepts it possesses, and making conjectures to connect some of these mathematical entities.

The program began initially with a collection of one hundred concepts selected from finite set theory, and in a couple hours it had defined about three hundred new concepts, half of which were quite well known in mathematics. One of the synthesized concepts was equivalent to natural numbers. AM rated this highly and spent much time developing elementary number theory, including conjecturing the fundamental theorem of arithmetic (each number has a unique prime factorization). This is of course much better behavior than one could expect from blind search through the space of legal mathematical definitions and propositions. In AM search is not blind; At any moment it can justify its current efforts merely by printing out the symbolic reasons for the task it is working on.

The design of AM is a blend of four powerful methods: frame representation, *heuristic search*, production systems, and *best-first search*. The concepts that AM discovers and explores are represented as frames (see article Representation.C7), each containing slots that are appropriate to the type of concept. For example, mathematical *operations* such as Addition have a Domain/Range slot that would be absent in frames that represent mathematical *objects* like Sets or Bags. The goal of AM is to develop its knowledge of mathematics by filling in empty slots in a concept and, occasionally, by defining new concepts. These tasks are suggested and performed by heuristic rules, represented as productions (see article Representation.C3). AM is constrained by these rules to explore potentially interesting concepts and aspects (slots) of concepts. After a heuristic has suggested that a slot be filled or a concept created, the suggested task must compete with others on an *agenda*, a job-list of plausible tasks. Each task is supported by a set of symbolic reasons and has a numeric weight representing its "interestingness." At each moment, AM directs its attention to the task with the highest weight.

The significance of the project lies both in the architecture of the program and in the fact that the program behaves well: AM is an existence proof that open-ended math research--theorem *proposing* not theorem proving--can be adequately represented (and automated) as a heuristic search. It is worth noting that the ultimate impediment to AM's progress was its inability to discover new heuristic rules, as it had discovered new mathematical concepts. By constructing and experimenting with the program it became clear where the next research thrust should be: along the direction of automating the discovery and evaluation of heuristics.

In the rest of this article, the nature of mathematical discovery is discussed. These ideas are then carried over into a description of the design of AM. The methods of knowledge representation and control are covered in depth, and special attention is given to the tasks that AM performs. An excerpt from a sample run of AM is given illustrating its discovery of prime numbers and perfect squares. Finally, AM is evaluated as a mathematician, and its limitations are noted.

A Model of Mathematical Research

Lenat's thesis was concerned with the mechanization of a particular type of mathematical activity (apart from theorem proving): the definition of new concepts and the recognition of plausible conjectures. The AM system has no proof capabilities. Below is the model of math research that AM was based upon, pieced together from the writings of Poincare, Polya, Lakatos, Hadamard, and others:

1. The order in which a math textbook presents a theory is almost the exact opposite of the order in which it was actually discovered and developed. In such a text, new definitions are presented as they are needed, with little or no motivation to state the next big theorem, whose proof then magically appears. In contrast, a mathematician doing research examines some already known concepts and tries to find some regularity in experimental data involving them. The patterns that he notices are the conjectures that he must investigate further, and these relationships directly motivate him to make new definitions.

2. Each step that the researcher takes while developing a new theory involves choices from a large set of "legal" alternatives. The key to keeping the search from becoming blind and explosive is the proper use of evaluation criteria. Each mathematician uses his own personal heuristics to choose the "best" alternative available at each moment.

3. Non-formal criteria (aesthetic interestingness, inductive inference from empirical evidence, analogy, and utility) are much more important than formal deductive methods in developing mathematically worthwhile theories, and in avoiding barren diversions.

4. It is sufficient, and pragmatically necessary, to have and use a large set of informal heuristic rules that direct the sequence of the researcher's activities, depending on the current situation. In addition, these rules can be assumed to superimpose: The combined effect of several rules is just the sum of the individual effects.

5. The necessary heuristic rules are virtually the same in all branches of mathematics and at all levels of sophistication. Each specialized field will have some of its own heuristics; those are normally much more powerful than the general-purpose heuristics.

6. For true understanding, the researcher should grasp--that is, have access to, relate to, store, be able to manipulate, be able to answer questions about, etc.--each concept in several ways, declaratively, abstractly, and operationally, and should know its relevance and examples of it.

Discovery in Mathematics

Before discussing the *synthesis* a new mathematical theory, we consider briefly its *analysis*, or how to construct a plausible chain of reasoning that stretches from a given discovery all the way back to well-known concepts.

One can rationalize a given discovery by working backwards, by reducing the creative act to simpler and simpler creative acts. For example, consider the concept of prime numbers. How might one be led to define such a notion if one had never heard of it before? Consider the following plausible strategy:

> If f is a function which transforms elements of A into elements of B, and B is ordered, then consider just those members of A which are transformed into *extremal* elements of B. This set is an interesting subset of A. Name it and study it.

When f(x) means "divisors of x" and the ordering is "by length," this heuristic directs one to consider those numbers that have a minimal number of factors--that is, the primes. So this rule actually *reduces* our task from proposing the concept of prime numbers to two more elementary problems: (a) discovering ordering-by-length and (b) inventing divisors-of.

Now suppose we know this general rule: "*If f is an interesting function, consider its inverse.*" It reduces the task of discovering divisors-of to the simpler task of discovering multiplication. Eventually, if followed far enough, this task reduces to the discovery of very basic notions like substitution, set-union, and equality. To explain how a given researcher might have made a given discovery, such an analysis must be continued until the inductive task is reduced to "discovering" the notions that the researcher started with, which were his conceptual primitives.

Syntheses of Discoveries

Suppose a large collection of these heuristic strategies has been assembled (e.g., by analyzing a great many discoveries and writing down new heuristic rules whenever necessary). Instead of using them to *explain* how a given idea might have evolved, one can imagine starting from a basic core of knowledge and "running" the heuristics to *generate* new concepts. It is simply the reversal of the process described in the last section: not *explanation*, but *generation*.

Notice that this *forward search* is much "bushier"--i.e., more branches or paths to follow--and much more explosive than the backwards analysis previously described. It is a much harder task to actually make a discovery than to rationalize--by hindsight--one already made.

Unconstrained forward search is too explosive (see Combinatorial Explosion in article Search.Overview); thus, we can hypothesize that the scientist employs some kind of informal rules-of-thumb or heuristics to constrain it. That is, he doesn't really follow rules like "*Look at the inverse of each known function f*", because that would take up too much time. Rather, his heuristic rules might be more naturally stated as productions (condition/action rules) like: "*If f is 1-1 and Range(f) << Domain(f), Then look at f-inverse.*" Henceforth, heuristic rule will mean such a conditional rule-of-thumb. In any particular situation some subset of these rules will "trigger" and suggest a manageable space of plausible activities to perform. After exploring that space for a while, the situation will have changed and the cycle will begin anew.

Design of the AM Program

Mathematical inductive syntheses are precisely what AM does. The program consists of a large corpus of primitive mathematical concepts, each with a few associated heuristics. Each such heuristic is a situation-action rule that functions as a local plausible move generator. Some suggest tasks for the system to carry out, some suggest ways of satisfying a given task, etc. AM's activities all serve to expand AM itself, to enlarge upon a given body of mathematical knowledge. AM uses its heuristics as judgmental criteria to guide development in the most promising direction.

Representation. Each concept is represented as a frame-like data structure with 25 different facets or slots. The types of facets include: EXAMPLES, DEFINITIONS, GENERALIZATIONS, DOMAIN/RANGE, ANALOGIES, INTERESTINGNESS, CONJECTURES and many others. Modular representation of concepts provides a convenient scheme for organizing the heuristics; for example, the following strategy fits into the *EXAMPLES* facet of the *PREDICATE* concept:

> If, empirically, 10 times as many elements FAIL some predicate P, as SATISFY it, then some *generalization* (weakened version) of P might be more interesting than P.

AM considers this suggestion after trying to fill in examples of each predicate. In fact, after AM attempts to find examples of *SET-EQUALITY*, so few are found that AM decides to generalize that predicate. The result is the creation of several new predicates, one of which happens to mean "Has-the-same-length-as," that is, a rudimentary precursor to natural numbers.

Below is part of a typical concept, PRIMES, in a state long after AM defined and explored it.

```
NAME: Prime Numbers
DEFINITIONS:
     ORIGIN: Number-of-divisors-of(x) = 2
     PREDICATE-CALCULUS: Prime(x) = (∀z)(z|x ⊃ z=1 ||| z=x)
     ITERATIVE: (for x>1): For i from 2 to Sqrt(x), \(i|x)

EXAMPLES: 2, 3, 5, 7, 11, 13, 17
               BOUNDARY: 2, 3
               BOUNDARY-FAILURES: 0, 1
               FAILURES: 12

GENERALIZATIONS: Nos., Nos. with an even no. of divisors, Nos. with a
                 prime no. of divisors

SPECIALIZATIONS: Odd Primes, Prime Pairs, Prime Uniquely-addables

CONJECS: Unique factorization, Goldbach's conjecture, Extremes of
         Number-of-divisors-of

ANALOGIES:
        Maximally divisible numbers are converse extremes of
            Number-of-divisors-of
        Factor a non-simple group into simple groups

INTEREST: Conjectures associating Primes with TIMES, and with Divisors-of

WORTH: 800
```

Creating a new concept is a well-defined activity: it involves setting up a new data structure like the one above and filling in entries for some of its slots. Filling in a particular slot of a particular concept is also quite well defined and is accomplished by executing a collection of relevant heuristic rules.

Control. AM is initially given a collection of 115 core concepts, with only a few slots filled in for each. Its sole activity is to choose some slot of some concept and fill in that particular slot. In so doing, new notions will often emerge. Uninteresting ones are forgotten, mildly interesting ones are kept as parts of one slot of one concept, and very interesting ones are granted full concept-module status. Each of these new modules has dozens of blank slots, hence the space of possible actions (blank slots to fill in) grows rapidly. The same heuristics are used both to suggest new directions for investigation and to limit attention, that is, both to sprout and to prune tasks on the agenda.

The fundamental kind of task that AM performs, is filling in a given facet of a given concept. To decide which such task to work on next, AM maintains an agenda of tasks, a global job-list ordered by priority. A typical task is "*Fill-in examples of Primes*". The agenda may contain hundreds of entries such as this one. AM repeatedly selects the top task from the agenda and tries to carry it out. In addition, AM creates plausible new tasks to place on the agenda and decides which task will be the best to execute next and how to carry it out.

If the task is "*Fill in new Algorithms for Set-union*", then *satisfying* it would mean actually synthesizing some new procedures, some new LISP code capable of forming the union of any two sets. A heuristic rule is *relevant* to a task if and only if executing that rule brings AM closer to satisfying that task. Relevance is determined a priori by where the rule is stored. A rule stored with the Domain/Range facet of the Compose concept would be presumed relevant to the task "*Check the Domain/Range of Insert-o-Delete*".

Once a task is chosen from the agenda, AM gathers some heuristic rules that might be relevant to satisfying that task. They are executed, and then AM picks a new task. While a rule is executing, three kinds of actions or effects can occur:

1. Facets of some concepts are filled in (e.g., examples of primes may actually be found and added to the "Examples" facet of the "Primes" concept). A typical heuristic rule that might have this effect is:

> To fill in examples of X, where X is a kind of Y (for some more general concept Y), check the examples of Y; some of them may be examples of X as well.

For the task of filling in examples of Primes, this rule would have AM notice that Primes is a kind of Number and therefore have it look over all the known examples of Number. Some of those would be primes and would be transferred to the Examples facet of Primes.

2. New concepts can be created (e.g., the concept "primes which are uniquely representable as the sum of two other primes" may somehow be deemed worth studying). A typical heuristic rule that might result in this new concept is:

> If some (but not most) examples of X are also examples of Y (for some concept Y), create a new concept defined as the intersection of those 2 concepts (X and Y).

Suppose AM has already isolated the concept of being representable as the sum of two primes in only one way (AM actually calls such numbers "Uniquely-prime-addable numbers"). When AM notices that some primes are in this set, the above rule will create a brand new concept defined as the set of numbers that are both prime and uniquely prime addable.

3. New tasks can be added to the agenda (e.g., the current activity may suggest that the following task is worth considering: "Generalize the concept of prime numbers"). A typical heuristic rule that might have this effect is:

> If very few examples of X are found, then add the following task to the agenda: "Generalize the concept X."

Of course, AM contains a precise meaning for the phrase "very few." When AM looks for primes among examples of already known kinds of numbers, it will find dozens of nonexamples for every example of a prime that it uncovers. "Very few" is thus naturally implemented as a statistical confidence level.

The concept of an agenda is certainly not new. *Schedulers* utilizing this concept have been around for a long time. But one important feature of AM's agenda scheme *is* a new

idea: attaching to each task a list of quasi-symbolic reasons that explain why the task is worth considering, why it's plausible. *It is the responsibility of the heuristic rules to include reasons for any tasks they propose.* For example, reconsider the heuristic rule mentioned in (3) above. It actually looks more like the following:

> If very few examples of X are found, then add the following task to the agenda: "Generalize the concept X," for the following reason--"X's are quite rare; a slightly less restrictive concept might be more interesting."

If the same task is proposed by several rules, then several different reasons for it may be present. In addition, one ephemeral reason also exists: Focus of attention. Any tasks that are similar to the one last executed get "Focus of attention" as a bonus reason. AM uses all these reasons to decide how to rank the tasks on the agenda. The "intelligence" AM exhibits is not so much "what it does" as the *order* in which it arranges its agenda. For example, in an experiment carried out with AM in which a randomly chosen task was alternated with the "best" task (the one AM chose to do), the system was only slowed down by a factor of 2; yet this behavior totally destroys its credibility as a rational researcher, as judged by the human user of AM.

AM uses the list of reasons in another way: Once a task has been selected, the quality of the reasons is used to decide how much time and space the task will be permitted to absorb, before AM quits and moves on to a new task.

A crucial heritability property holds: Any method for filling in facet f of concept C will also work for filling in facet f of any *specialization* of C. Thus, when the task "Fill in examples of *SET-EQUALITY*" is chosen, AM asks each *generalization* of *SET-EQUALITY* for help. It asks for ways to fill in examples of any Predicate, any Activity, any Concept, and finally of Anything. One such heuristic rule known to the Activity concept says: "Actually execute the activity on some random members of its domain." Hence, to fill in examples of *SET-EQUALITY*, its domain facet is inspected, and AM sees that it takes a pair of objects as its arguments. Then AM accesses the Examples facet of the concept *OBJECT*, where it finds a large list of objects. Obeying the heuristic rule, AM repeatedly picks a pair of objects at random and sees if they satisfy *SET-EQUALITY* (by actually running the LISP function stored in the Algorithms facet of *SET-EQUALITY*). While this step will typically return False, it will occasionally locate--by random chance--a pair of equal sets.

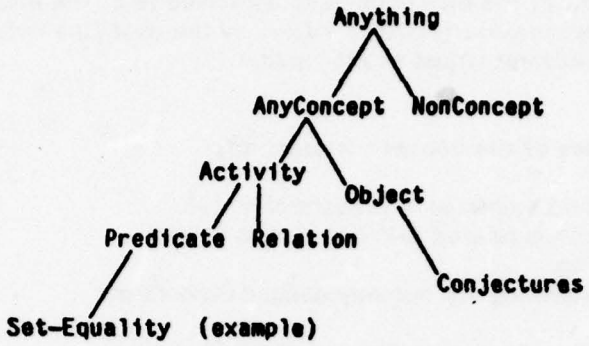


Figure 1. Partial path of property inheritence for Concepts in AM

Other heuristics, added to other generalizations of *SET-EQUALITY*, provide additional methods for executing the task "Fill in examples of *SET-EQUALITY*." A heuristic stored on the concept *ANY-CONCEPT* says to symbolically instantiate the definition. A bag of tricks is provided for this purpose, one of which ("Instantiate the base step of the recursion") works nicely on the recursive definition provided for *SET-EQUALITY*. Notice that, as one might expect, the more general the concept is, the weaker (more time-consuming) its heuristics are. For this reason, AM consults each concept's rules in order of increasing generalization.

Executing a task is achieved by locating relevant rules-of-thumb and evaluating them. The location is performed efficiently because all the concepts are related by generalization/specialization links and because the above "heritability" property holds.

Notice the omnipresent reliance upon heuristic guidance. Heuristics propose the tasks (and associate reasons for them) for the agenda; they propose new concepts to be defined; they discover (by search, synthesis, or analysis) entries that can be put into specific facets of specific concepts. There are even heuristics for naming new concepts (based on how they were formed).

An Excerpt

To convey a bit of AM's flavor, we present a brief excerpt of it in action. After reading through it, the reader should be convinced that AM is neither proving theorems nor *randomly* manipulating entries in a knowledge base, nor is it *exhaustively* manipulating or searching. AM is carefully growing a network of data structures representing mathematical concepts, by repeatedly using heuristics both (a) to guide the choice of what task to work on next and (b) to provide methods to satisfy the chosen task.

Although AM appears to have reasonable natural language abilities, very little effort was expended in this area. All text was generated using simple, stored patterns associated

with the heuristic rules. A few of the lines have been anglicised (e.g., the addition of "e.g."), and some of the concepts had previously been renamed by the user (the human observor of the system). Otherwise, the excerpt is just as AM typed it:

TASK 65: Fill in Examples of the concept Divisors-of.

3 Reasons: (1) No known examples of Divisors-of so far.
(2) TIMES, which is related to Divisors-of, is now very interesting.
(3) Focus of attention: AM recently defined Divisors-of.

26 examples found, in 9.2 seconds. E.g., Divisors-of(6)={1 2 3 6}.

TASK 66: Consider numbers having small sets of Divisors-of.

2 Reasons: (1) Worthwhile to look for extreme cases.
(2) Focus of attention: AM recently worked on Divisors-of.

Filling in examples of numbers with 0 divisors.
0 examples found, in 4.0 seconds.
Conjecture: no numbers have precisely 0 divisors.

Filling in examples of numbers with 1 divisors.
1 examples found, in 4.0 seconds. e.g., Divisors-of(1) = {1}.
Conjecture: 1 is the only number with precisely 1 divisor.

Filling in examples of numbers with 2 divisors.
24 examples found, in 4.0 seconds. e.g., Divisors-of(13) = {1 13}.
No obvious conjecture. May merit more study.
Creating a new concept: "Numbers-with-2-divisors".

Filling in examples of numbers with 3 divisors.
11 examples found, in 4.0 seconds. e.g., Divisors-of(49) = {1 7 49}.
All numbers with 3 divisors are also Squares.
Definitely merits more study.
Creating a new concept: "Numbers-with-3-divisors".

TASK 67: Consider the square-roots of Numbers-with-3-divisors.

2 Reasons:
(1) Numbers-with-3-divisors are unexpectedly also Perfect Squares.
(2) Focus of attention: AM recently worked on Nos-with-3-divisors.

All square-roots of Numbers-with-3-divisors seem to be Numbers-with-2-divisors.
E.g., Divisors-of(Square-root(169)) = Divisors-of(13) = {1 13}.

Even the converse of this seems empirically to be true.

I.e., the square of each No-with-2-divisors seems to be a No-with-3-divisors.
The chance of coincidence is below acceptable limits.

Boosting the interestingness rating of each of the concepts involved.

USER: Rename Numbers-with-2-divisors as Primes

TASK 68: Consider the squares of Numbers-with-3-divisors.

3 Reasons: (1) Squares of Numbers-with-2-divisors were interesting.
(2) Square-roots of Numbers-with-3-divisors were interesting.
(3) Focus of attention: AM recently worked on Nos-with-3-divisors.

Results: AM as a Mathematician

Here we will review the mathematics that AM explored. Thoughout, AM acted alone, with a human user watching it and occasionally renaming some concepts for his or her own benefit. Like a contemporary historian summarizing the work of the Babylonian mathematicians, current terms are used and criticism is by current standards.

AM began its investigations with scanty knowledge of a few set-theoretic concepts. Most of the obvious set-theory relations (e.g., de Morgan's laws) were eventually uncovered; since AM never fully understood abstract algebra, the statement and verification of each of these was quite obscure. AM never derived a formal notion of infinity, but it naively established conjectures like "a set can never be a member of itself," and procedures for making chains of new sets ("insert a set into itself"). No sophisticated set theory (e.g., diagonalization) was ever done.

After this initial period of exploration, AM decided that "equality" was worth generalizing and thereby discovered the relation "same-size-as." Natural numbers were based on this discovery, and, soon after, most simple arithmetic operations were defined.

Since addition arose as an analog to union, and multiplication as a repeated substitution, it came as quite a surprise when AM noticed that they were related (namely, $N + N = 2 \times N$). AM later rediscovered multiplication in three other ways: as repeated addition, as the numeric analog of the Cartesian product of sets, and using the cardinality of the power set of the union of two sets.

Raising to fourth-powers and fourth-rooting were discovered at this time. Perfect squares and perfect fourth-powers were isolated. Many other numeric operations and kinds of numbers were found to be of interest: Odds, Evens, Doubling, Halving, Integer-square-root, etc. Although it isolated the set of numbers that had no square root, AM was never close to discovering rationals, let alone irrationals. No notion of "closure" was provided to--or discovered by--AM.

The associativity and commutativity of multiplication indicated to AM that it could accept a BAG of numbers as its argument. When AM defined the inverse operation corresponding to "Times", this property allowed the definition to be: "Any bag of numbers (>1) whose product is x." This was just the notion of factoring a number x. Minimally factorable numbers turned out to be what we call primes. Maximally factorable numbers were also thought to be interesting.

Prime pairs were discovered in a bizarre way: by restricting the domain and range of addition to primes (i.e., solutions of p + q = r in primes).

AM conjectured the fundamental theorem of arithmetic (unique factorization into primes) and Goldbach's conjecture (every even number >2 is the sum of two primes) in a surprisingly symmetric way. The unary representation of numbers gave way to a representation as a bag of primes (based on unique factorization), but AM never thought of exponential notation. Since the key concepts of remainder, greater-than, gcd, and exponentiation were never mastered, progress in number theory was arrested.

When a new base of *geometric* concepts was added, AM began finding some more general associations. In place of the strict definitions for the equality of lines, angles, and triangles came new definitions of concepts comparable to Parallel, Equal-measure, Similar, Congruent, Translation, Rotation; together with many that have no common name (e.g., the relationship of two triangles sharing a common angle). A cute geometric interpretation of Goldbach's conjecture was found: Given all angles of a prime number of degrees, (0,1,2,3,5,7,11,...,179 degrees), then any angle between 0 and 180 degrees can be approximated (to within 1 degree) as the sum of two of those angles. Lacking a geometry "model" (an analogic representation like the one Gelernter, 1963 employed), AM was doomed to propose many implausible geometric conjectures (see Artilce Representation.C6).

It is important to ask how general the program is: Is the knowledge base "just right" (i.e., finely tuned to elicit this one chain of behaviors)? The answer is no: The whole point of this project was to show that a relatively small set of general heuristics can guide a nontrivial discovery process. Keeping the program general and not finely tuned was a key objective. Each activity or task was proposed by some heuristic rule (like "look for extreme cases of X") that was used time and time again, in many situations. It was not considered fair to insert heuristic guidance that could only "guide" in a single situation. For example, the same heuristics that lead AM to decompose numbers (using TIMES-inverse) and thereby discover unique factorization, also lead to decomposing numbers (using ADD-inverse) and the discovery of Goldbach's conjecture.

Results: Limitations of AM

Although AM fared well according to several different measures of performance, users of this handbook may better utilize knowledge of its *limitations*.

As AM ran longer and longer, the concepts it defined were further and further from the primitives it began with, and the efficacy of its fixed set of 250 heuristics gradually declined. The key deficiency was the lack of adequate *meta-rules* (Davis, 1976, Lenat, 1976, Laing, 1971): heuristics that could cause the creation and modification of new heuristics. This lack is strongly felt in a boot-strapping, open-ended task environment such as math research.

Many concepts that one might consider "primitive" are missing from AM: proof, tricks for finding counterexamples, numbers, etc. Very few of them are ever discovered by AM, and even those that are discovered will not have any powerful heuristics filled in for them. The limitations of a small knowledge base can be overcome only by investing additional time to enlarge it. With a learning system like AM, one can spend a couple man-hours wrestling with each new concept or let the program squander a greater amount of its time until it has discovered and mastered that concept to the same level of proficiency. It is a trade-off that almost always argues for the system-builder to spend more time enlarging the knowledge base by hand.

Analogies in general were underutilized. Specifically, analogies between heuristics were never even considered. If one characterizes an analogy as a (partial) correspondence between two collections of objects and operators, then it is a small conceptual step to imagine heuristic rules that look for and develop such mappings: The image of partial matching comes immediately to mind. AM possessed a few such domain-independent rules, and they managed to produce some analogies (e.g., between multiplication and addition; between sets/union/same-size and numbers/add/equality), but the overall results were quite meager in this area.

Conclusions

The AM project stands as a working demonstration that a few hundred general heuristic rules suffice to guide a searcher--an automated math researcher--as it explores and expands a large but incomplete base of math concepts. AM shows that creative theoretical research can be effectively modeled as heuristic search, just as Meta-Dendral (see article C2c) established a similar hypothesis for the more constrained, real-world field of mass spectroscopy.

The main successes were the few relatively novel ideas it came up with (including a result in number theory, dealing with numbers having very many divisors), the ease with which new domains were discovered (e.g., number theory) or introduced by hand (plane geometry), and the apparently rational sequences of behavior that AM exhibited.

The continuation of this line of research by Lenat is the EURISKO program. The hypothesis being explored is that the meta-level knowledge required to synthesize and reason about heuristics is a *subset* of the knowledge already in AM about synthesizing and reasoning about concepts. That is, EURISKO's meta-rules are merely some of the very general rules that AM already had. The only real change, then, from AM to EURISKO is to recode each heuristic from LISP code as a full-fledged concept with facets. The heuristics, which deal with facets of concepts, will then be capable of dealing with each other. This work is currently in progress at Stanford University.

Future AM-like programs may serve as assistants to scientists and engineers, synergetically collaborating with them in the conception, planning, and execution of their research and development activities.

References

See Lenat (1977). A much expanded description of the system is in the thesis by Lenat (1976).

D2. MACSYMA

MACSYMA is a large, interactive computer system designed to assist mathematicians, scientists, and engineers in solving mathematical problems. It has a wide range of algebraic manipulation capabilities, all working on symbolic inputs and yielding symbolic results, as well as an extensive numerical subroutine library (IMSL) and plotting package.

MACSYMA is used extensively by hundreds of researchers from government laboratories, universities, and private companies throughout the United States. Many of these users spend a substantial portion of every day logged in. Currently, the system runs exclusively on a Digital Equipment Corporation KL-10 at MIT and is accessed through the ARPA Network; however, there are plans to distribute it to other sites in the near future. MACSYMA's funding is supplied almost exclusively by its user community.

The original design for MACSYMA was laid out by Carl Engleman, Bill Martin, and Joel Moses in 1968. They built on their previous experience with the Mathlab 68 system and the theses of Martin and Moses. Martin had constructed an algebraic manipulation system to solve certain problems in applied mathematics. Moses had produced a program that was able to do indefinite integration about as well as a typical graduate student. The system had its first users in 1971 and has undergone continuous development since then, a total of about 45 man-years of effort.

The implementation of MACSYMA is based on the belief that the way to produce a high-performance program for general mathematics is to "build in" a large amount of knowledge. This approach to system construction is often called "knowledge-based programming." MACSYMA is an extremely large system, as algebraic manipulation systems go; at present, it can perform at least 600 distinct mathematical operations, including differentiation, integration, solution of equations and of systems of equations, Taylor series expansions, matrix operations, vector algebra, order analysis, etc. The current system consists of about 230,000 words of compiled LISP code and an equal amount of code written in the MACSYMA programming language. About half of this code was written by MACSYMA staff members; the rest was contributed by various users.

The primary goal of algebraic manipulation research has been the invention and analysis of new mathematical algorithms and the extension of previously known numerical algorithms to symbolic manipulation.

While most of the algorithms incorporated in MACSYMA were known to mathematicians prior to its construction, a substantial number came about as a result of this research. The last decade has brought the discovery of new algorithms for finding the greatest common divisors of polynomials (Brown and Traub, 1971; Moses and Yun, 1973), factoring rational expressions (Musser, 1975; Wang and Rothschild, 1975), sum simplification (Gosper, 1977), symbolic integration (Moses, 1971; Norman, 1975; Risch, 1969; Rothstein, 1977; Trager, 1978), and asymptotic analysis (Fateman, 1976; Norman, 1975; Zippel, 1976). The nature of this work has been largely mathematical; and, although Artificial Intelligence was instrumental in providing the environment in which MACSYMA was created, it has made little direct contribution since then.

Knowledge-based programming does, however, engender a number of difficulties for which AI techniques offer partial answers. Two general types of difficulties are discussed

here, namely, user education and the handling of mathematical problems not amenable to algorithmic solution.

Non-algorithmic procedures in MACSYMA

One of the most pressing problems in algebraic manipulation is simplification. Symbolic algorithms often generate large, unwieldy expressions that must be simplified into smaller, more meaningful forms. (Generally, the size of expressions is the most important criterion for simplicity, with standard formats and particularly revealing forms taking precedence.) To help users simplify their results, MACSYMA provides a variety of explicit expression transformation commands (such as expansion, factorization, partial fraction decomposition, etc.) and a simplifier that automatically applies a set of mathematical "rules" to every new expression as it is constructed. Examples of these rules are:

$$x*x \rightarrow x^2$$
$$\sin(x+\pi/2) \rightarrow \cos(x)$$
$$\log(a*b) \rightarrow \log(a)+\log(b) \ .$$

The user can, of course, define new commands and new rules.

Semantic Pattern Matching

In applying a simplification rule, MACSYMA utilizes a "semantic pattern matcher" to find instances of the rule's pattern. The matcher is "semantic" in that it uses knowledge about the operators and constants in an expression to find nonsyntactic matches. For example, the pattern $a*x^2 + b*x + c$, where a, b, and c are pattern variables free of x, will match the expressions $4*x^2 + 4*x + 1$, $x^2 + x + 1$, x^2, and $(x + 1)^2$. In defining a rule, the user may specify arbitrary conditions (in the form of procedural predicates) on the pattern variables. For example, determining whether an expression matches the above pattern, MACSYMA would call a user-specified function to check that any tentative assignments for a, b, and c are free of x. As a result, the pattern would not match $4*x^2 + 3*x + \sin(x)$.

One problem with this pattern matcher is that the user is unable to control how much "semantics" the system uses in finding a match. In the very near future, a new pattern matcher will be released in which the user will be able to specify a set of identities to use in attempting to identify instances of patterns. For example, while it is often desirable that the matcher use inverses, in some situations a user might prefer a simpler matcher, lest the rule a*b -> c apply to every lone a and b, as in b -> c/a. With the new pattern matcher, the user will be able to specify when he wants the inverse axioms to be used.

Simplification by Hillclimbing

While size of an expressions is not the sole criterion for its simplicity, it is a useful guideline. For those applications in which the user desires the smallest possible form for an expression, MACSYMA provides a search-oriented simplifier called SCSIMP. Given an expression and a set of rules, SCSIMP applies each of the rules to the expression, in turn, and retains the smallest result. If any such substitution leads to an expression smaller than

the original, the process is repeated. For example, given the identities below, SCSIMP will convert the first expression into the last.

$$K^2 + L^2 = 1 \qquad\qquad N^2 - M^2 = 1$$

First expression: $K^2 N^2 + K^2 M^2 N^2 - K^2 L^2 N^2 - K^2 L^2 M^2 N^2$

Intermediate: $K^4 M^2 N^2 + K^4 N^2$ substituting for L

Final Expression: $K^4 N^4$ substituting for M

Note, however, that because SCSIMP is a hillclimbing algorithm it is not guaranteed to produce the smallest answer. For example, it would not perform the simplification shown below.

First expression: $K^2 N^2 + L^2 M^2$

Intermediate form: $K^2 N^2 - K^2 M^2 + M^2$ substituting for L

Simplest form: $K^2 + M^2$ substituting for N

The reason for not performing this simplification is that in order to arrive at the simplest form, a larger intermediate expression would have to be generated. Due to the combinatorics involved in generating arbitrarily large intermediate forms, this technique has not been incorporated in the current version of SCSIMP.

The Relational Database and Inference

In certain problems, the symbols in mathematical expressions have restrictions on their ranges or on other properties that are useful in simplification. In order to allow the user to specify such properties, MACSYMA maintains a *relational database* of facts about symbols, stored in the form of a semantic network. For example, a user can declare (via the DECLARE command) that the symbol n is restricted to integer values, and MACSYMA can then simplify cos((2*n + 1)*π) to 0. Similarly, one can specify (via the ASSUME command) that x <= y, y <= z, and z <= x; and MACSYMA can then deduce that x = y = z (using the algorithm described below).

The database retrieval routines are supplemented by a fast but limited inference algorithm called CPM (Genesereth, 1976), which performs taxonomic deductions, property inheritances, set intersections, and other simple inferences. For example, given the facts that X is an integer, that integers are rational, and that the real numbers are partitioned into rationals and irrationals, CPM automatically deduces that X is not an irrational. Given the fact that a rational can be written as an integral numerator over an integral denominator,

CPM automatically deduces that X can be so written. The CPM inference algorithm was developed to enhance the retrieval capabilities of a high-level database system organized as a semantic network. It is an elaboration of Grossman's work (Grossman, 1976) on *constraint expressions* but has been carefully restricted so as to be susceptible to implementation on parallel hardware. The algorithm is a highly "compiled" form of *domain-independent constraint propagation* in which constraints, represented by "labels" on the nodes of the network, propagate across links to other nodes according to the laws of logic. It can perform certain inferences much more efficiently than their straightforward implementation in procedural problem-solving languages like CONNIVER. For further details on the CPM algorithm, the reader should consult Genesereth, 1976. In addition, Fahlman (1977) has described how such a constraint propagation algorithm can be implemented in parallel hardware for even greater efficiency.

Heuristic Problem Solving

MACSYMA also includes a number of specialized procedural problem solvers; for example, the first phase of the integration routine (Moses, 1971), the commands for performing root contraction and logarithmic contraction, the inequality theorem prover, and others.

User Education

The advantage of a large knowledge-based system like MACSYMA over a smaller, sparer system like REDUCE (Hearn, 1973) is that MACSYMA has more mathematical knowledge built in (i.e., it is larger and can do more). As a consequence, the user is not forced to communicate as much mathematical knowledge to the system. The disadvantage is that MACSYMA can be more difficult to understand and to use. The user might, for example, be unaware of the capabilities available or not know the commands, or he might get an unexpected result that he cannot explain.

To minimize these difficulties, MACSYMA offers a wide range of on-line user aids (Genesereth, 1977; Lewis, 1977), including a frame-oriented interactive primer (similar to PLATO), an information network, and an automatic program for searching the reference manual. In addition, some of MACSYMA's commands are able to explain their progress in a fashion that can be comprehended by the user. For example, if the VERBOSE option is selected, the POWERSERIES command prints out the goals and subgoals that it generates while working on an expansion.

Even with these provisions, users occasionally encounter difficulties due to their lack of knowledge of the system. Furthermore, such users are often unwilling to learn more about MACSYMA than is necessary to solve an immediate problem. The simplest way for such a user to acquire just the information he needs is to ask a consultant for help. Then, armed with the consultant's advice, he can surmount the difficulty and solve the problem.

Consultation is a method widely used in computer centers as well as in domains like business, law, and medicine; and, as computer technology becomes more pervasive and computer systems become more complex, the need for consultation grows. Unfortunately, human consultants are a scarce resource and quite expensive. Currently, work is underway

on an automated consultant for MACSYMA novices, called the Advisor. It is a program distinct from MACSYMA, with its own database and expertise. The Advisor accepts a description of a difficulty from its user and tries to reconstruct the user's "plan" for solving his problem. Based on this plan and its knowledge of MACSYMA, the Advisor then generates advice tailored to the user's specific need. For a description of the Advisor's operation, the user should see Genesereth, 1978.

Future Plans

In addition to the features described above, several other AI-related capabilities are under development in MACSYMA. Two of these are mentioned here, namely a new representation for algebraic expressions using data abstractions and a knowledge-based, plan-based mathematician's (or physicist's or engineer's) "apprentice."

Recently, David Barton has designed a radically new scheme for representing algebraic expressions. MACSYMA has two major representations, the general representation that uses LISP's traditional prefix format and the rational representation that uses a canonical form for polynomials and rational functions. The rational representation has become unwieldy over the years, as extensions to the system have changed its specifications. For example, coefficients of polynomials were originally assumed to be integers and were later generalized to include floating point numbers. A new representation was desired to handle "Taylor series," which contains rational number exponents, since the former representation, while relatively close to the rational representation, could not be retrofitted onto the rational representation. Barton's approach alleviates these difficulties and provides a capability for future generalization. The approach used is, furthermore, a natural one for abstract algebra.

Consider, for example, a 2 X 2 matrix whose elements are Laurent series in y (truncated at y^2), whose coefficients are polynomials in x, whose coefficients are rational numbers. In order to add such a 2 X 2 matrix to another 2 X 2 matrix, one needs to know how to add the elements. One approach would be to design a general addition routine that would check the types of each argument and finally perform the appropriate addition. This approach is similar to the one previously taken by the rational function representation. In a symbolic system, and, in fact, in most applications, the type of object is intimately related to a set of operations that can be performed on it. In the MACSYMA context, these operations include addition, subtraction, multiplication, division, differentiation, substitution, coefficient extraction, and GCD computation. Barton's approach is to attach a tree of vectors to each expression. The tree corresponds to the gross structure of the expression. For example, each subexpression, an element in the matrix, has a vector corresponding to it. The vector's elements are in a fixed order and contain pointers to the procedures that perform the corresponding operation on the type of the subexpression.

Barton's approach permits expressions to be composed of arbitrarily nested types. This is a critical requirement in an interactive symbolic system. Preliminary tests of expressions represented in this manner indicate that common manipulations are not much slower and often faster than in the former implementation. The reason for a speed-up is that less type-testing is needed in this approach.

Work has also begun on the design of an "apprentice" for the MACSYMA user. At present, MACSYMA is used mostly as a "symbolic calculator," with the user directing its

actions line by line and keeping track of the meaning of each result. The goal of the apprentice is to relieve the user of much of this drudgery. The approach being taken involves two components, namely knowledge about the user's domain and the use of a high-level problem-solving plan formalism.

Currently, most symbols in MACSYMA have no special meaning, and they can take on arbitrary values. In particular problem areas, however, certain symbols have particular interpretations and range restrictions. For example, the symbol MASS has a very special meaning to physicists and an obvious range restriction (nonnegative). A physicist's apprentice should know this range restriction and be able to use it; for example, in discarding negative roots or performing integrations. Similarly, practitioners in certain fields like to see their expressions written in standard formats, determined by the interpretation of the constituent symbols. For example, electrical engineers usually prefer resistance (Ri) and capacitance (Ci) expressions written as f(R1, R2, ..., Rn)*g(C1, C2, ..., Cn) rather than having the Ri and Ci intermixed.

Another way that an apprentice could be of use in MACSYMA is by keeping track of the user's *plan* for solving his problem. If the apprentice knows the steps involved and the significance of various results, it could inform the user of potential errors, make suggestions, and in many cases carry out steps by itself. The apprentice can gain familiarity with the user's plan in various ways: it may be a well-known mathematical procedure (e.g., some standard technique for solving partial differential equations or perturbation problems), the user may have described his intentions before beginning his MACSYMA session, or the user may re-apply some previous plan. It is expected that this notion of a problem-solving plan will play an extremely important role in the next generation of algebraic manipulation systems.

References

Unfortunately, there is no good introductory reference on the structure of MACSYMA. The reader is referred to the MACSYMA manual (Mathlab Group, 1977) and the primer (Moses, 1975) for an introduction to its use.

See also Brown and Traub (1971), Fahlman (1977), Fateman (1972), Genesereth (1976), Genesereth (1977), Genesereth (1978), Gosper (1977), Grossman (1976), Hearn (1973), Lewis (1977), Moses (1971), Moses and Yun (1973), Musser (1975), Norman (1975), Risch (1969), Rothstein (1977), Trager (1976), Wang and Rothschild (1975), and Zippel (1976).

E. Other Scientific Applications

E1. The SRI Computer-based Consultant

A Computer-based Consultant (CBC) is a computer system that contains a body of specialized knowledge about a particular task domain and which makes that knowledge conveniently available to users working in the domain. This article describes some research done at SRI on a computer-based consultant designed to help a novice mechanic work with electromechanical equipment. The goal of this research is to build a system that approximates a human consultant in its communication, perceptual, and reasoning skills.

The consultant was designed to answer spoken English questions from the user and to monitor the user's progress on the task, offering advice and reminders where necessary. To fit the needs of individual users, it is essential that the system be able to provide advice about the task at several levels of detail. In order to determine the appropriate level of detail, the CBC must form a model of the user, monitor his performance as he executes the task, and update internal models to reflect the current state of the task environment.

Design of the Computer-based Consultant.

The task of the SRI computer-based consultant is to help an inexperienced mechanic repair and modify complex electromechanical equipment. The mechanic works on a piece of equipment in a special "work station" where he is provided with a headset that enables him to talk to the system and to receive spoken replies, both in natural language. A commercially available phoneme synthesizer is used by the system to give "spoken" responses to the user, and a commercially available phrase recognizer is used to "understand" his speech. There is a television camera and a laser rangefinder that provide the visual component for the system. The laser rangefinder can also be used as a visual pointer so that the system can answer questions such as "Show me the pressure switch" by illuminating the pressure switch with the laser beam.

Requests for information by the user are translated into an internal representation or "model" by the natural language and visual components of the system. These models are used to structure communications with the user as he performs the task. For example, a question about the location of a part ("Where is the pump brace) is answered by reference to a stored geometric model that keeps track of the spatial relations between the parts. Other models are necessary for the natural language components of the system; for instance, a discourse model is needed to understand a spoken utterance.

Planning a sequence of constructions

The user of the CBC can ask it to plan a sequence of assembly steps and relate this sequence to him for execution. The CBC has a planning component for composing assembly and disassembly sequences. It has received much attention in recent research efforts. There are several types of knowledge that are important in the planning process. First, there is the model of the air compressor itself, which is essentially a graph whose nodes correspond to the parts of the compressor and whose arcs correspond to the mechanical connection between the parts. Second, each type of connection has associated with it a

set of procedures that tells how that connection is physically established. Third, each of these procedures may contain calls to other procedures that elaborate, in more detail, how a job is done. This *hierarchy of procedural knowledge* forms the basis for producing plans that can be given to the user at several levels of detail. This procedural model is used by the planning program to determine the order in which parts should be assembled. The planning program initially assumes that the parts can be connected in any order. By checking preconditions and the effects of performing each step, it reorders the steps in the plan to eliminate conflicts. For example, the pump can be installed only if there is no pulley on its shaft. The planner recognizes this fact and imposes an order on the plan so that the pump will be installed before its pulley is placed on the shaft. When all the conflicts have been resolved, the remaining steps of the plan can be solved in any order. This ability, to recognize alternative orderings in a plan, is important for a computer-based consultant: The user may take the initiative and proceed with certain steps of the assembly on his own, and the planner must recognize if the steps being taken are valid.

The plan is represented as a structure called a procedural net; a sample net is shown in Figure 1 (Hart, 1975). Each node corresponds to an assembly step at some level of detail. The procedural net is actually a hierarchy of plans, all of which accomplish the same task, but at varying levels of detail. The *i*th row in the net corresponds to a plan specified at the *i*th level of detail. Notice that the plan splits into two paths at level 2, indicating that the two subplans can be performed in either order. The branching verticle lines indicate the expansion of a step into a more detailed subplan.

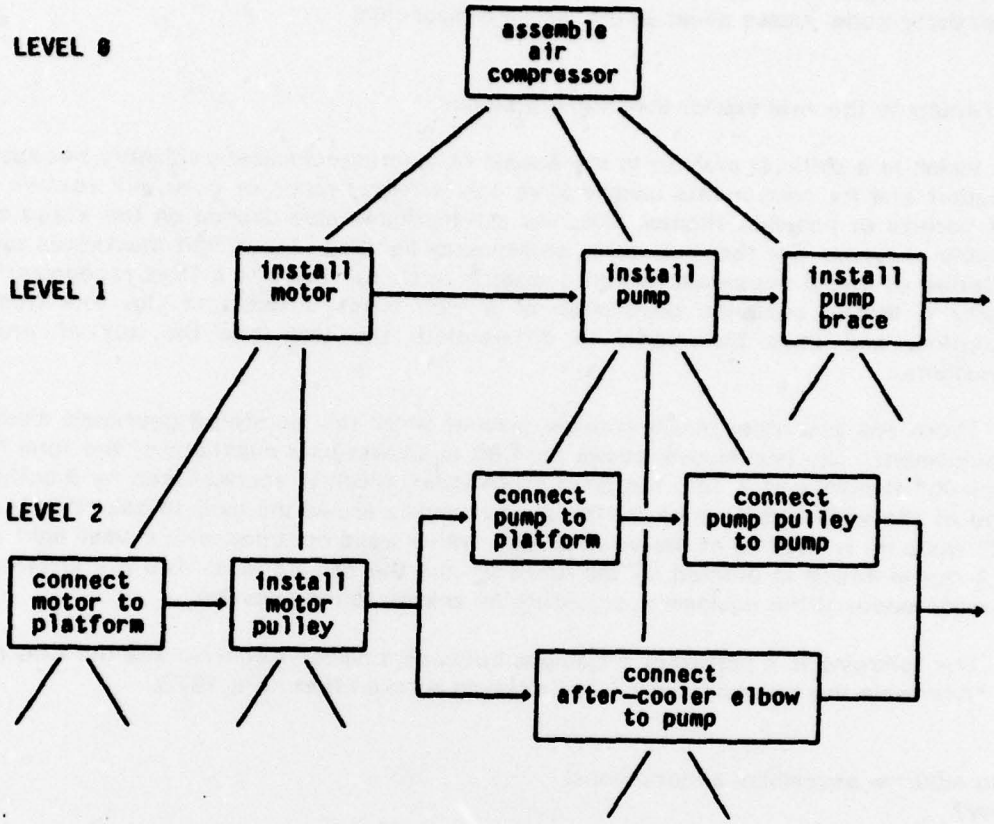


Figure 1. A fragment of a procedural net.

The procedural net is useful for the specification of plans at the various levels of detail required by the user. The net is also used during planning to represent partially formed plans, so that the planner can be restarted during execution to modify an existing plan if new information requiring replanning is discovered as the assembly proceeds.

The system keeps an internal model of the "connectedness" of the different components of the air compressor as it is being assembled or disassembled. It also contains a graphical model of the locations of the different components of the air compressor. For

example, the graphical model knows about the pump, motor, pressure switch, etc.; and the connectivity model knows about all the visible components.

Interfacing to the real world: Vision and Dialogue

Vision is a difficult problem in the domain of electromechanical machinery because the equipment and its components usually have only a limited range of color and texture for a great variety of possible shapes. Shadows and highlights also depend on the stage of the assembly process. For these reasons, an approach has been taken that capitalizes on prior knowledge of visual appearances and geometric relations: There is a "tool recognizer" that accepts a limited semantic description of a tool, builds a model of the tool from this description, and uses this model to differentiate the tool from the set of probable alternatives.

There are two other visual modules, both of which rely on stored geometric models of the equipment. The first module allows the CBC to answer user questions of the form "Show me the X." It points at X with the laser rangefinder, which is accomplished by locating the outline of the visible surface of X. The second module allows the user to ask "What part is this?" while he is pointing at the unknown part with a wand equipped with a small light at the tip. A ray in space is defined by the wand tip and the camera lens. The ray intersects a geometric model of the equipment, providing the answer to the question.

The following is a portion of a dialogue between a novice mechanic and the CBC on the task "Assemble the air compressor." The example is taken from Hart, 1975.

Try to achieve assembled aircompressor.
**How?
The top level plan is ... install pump ...
install pumpbrace ... install pumppulley ...
install belt ... install belthousingcover ...
Ready for questions.
**Show me the pump.

Laser beam moves to point at pump.

Ready for questions.

.
.
.

**What is this?

User points with wand at tank platform...

Tank platform.
Ready for questions.
**None.
Install pump.
**How?
Connect pump to platform.

**How?
Get 4 1&1/4-inch 5/16 hexagonal bolts and
4 9/16 flat washers and 4 size 5/16 hexagonal nuts.
**OK.

.
.
.

Summary

The main goal of the research on the SRI computer-based consultant was to create the fundamental technology needed to build such consultant systems. Much of this technology is independent of the particular consultant system being built or of the details of the task environment. Work on the electromechanical CBC described here was stopped in April, 1975. Since then a new consultant project has been begun at SRI extending many of the ideas from the 1975 system and adding new features.
The new consultant system is being used in two entirely different domains: mineral exploration (see article E2) and agricultural pest management. It appears that computer consultants are likely to remain both a fruitful area for Artificial Intelligence research and a practical means of deploying knowledge to people working in the specific task domains.

References

See Hart (1975) and Nilsson (1975).

E2. PROSPECTOR

PROSPECTOR (Duda et al., 1978) is a computer-based consultation system that is being developed to aid geologists working on certain problems in hard-rock mineral exploration. Like other expert systems, such as INTERNIST (see article Medicine.C4) and MYCIN (see article Medicine.C2), PROSPECTOR attempts to represent a significant portion of the knowledge and the reasoning processes used by experts working in a specialized domain. The intended user of this program is an exploration geologist who is in the early stages of investigating an exploration site or prospect. We assume that he or she has a professional understanding of geology but, nevertheless, wants the assistance of a specialist in evaluating the findings.

In an attempt to keep the PROSPECTOR system fairly domain independent, the system maintains a clear separation between the geological knowledge base and the mechanisms that use this knowledge. Nevertheless, characteristics of the problem domain have certainly influenced the design of the system.

Expressed abstractly, the main function of PROSPECTOR is to match data from a particular situation against "models" that describe a moderately large number of disjoint classes of situations. In PROSPECTOR's domain, the models are formal descriptions of the most important types of ore deposits, and the data are primarily surface geological observations. The available data are assumed to be uncertain and incomplete, so that the conclusion is a probability or a degree of match. In addition, the program alerts the user to different possible interpretations of the data and identifies additional observations that would be most valuable for reaching a more definite conclusion.

A typical consultation session with PROSPECTOR begins with the user giving the system the information about the most significant features of his prospect: the major rock types, minerals, and alteration products. The program matches these observations against its models and, when the user is finished volunteering information, proceeds to ask the user for additional information that will help confirm the best matching model. At any time during the consultation, the user can interrupt to volunteer new information, change previous statements, or request an evaluation. Thus, PROSPECTOR operates as a mixed-initiative system.

The following edited excerpt from a PROSPECTOR run illustrates many of these ideas. The data for the run describe a fictitious porphyry copper deposit that has some, but not all, of the desired characteristics. We begin at the point where the user starts volunteering information to the system. In this excerpt, the user's answers are always given in **emboldened** upper case, the system's comments are in regular type, and explanatory comments are in *italics*.

1: THERE IS QUARTZ MONZONITE

QUARTZ MONZONITE (5)

2: THERE IS A CRETACEOUS DIKE

CRETACEOUS DIKE (5)

3: THERE IS SERICITE AND PYRITE

SERICITE (5)
PYRITE (5)

5: THERE MAY BE BIOTITE

BIOTITE (2)

6: DONE

These simple assertions describe the most prominent rocks, minerals, and alteration products in the target area. The system indicates its recognition of each statement by echoing the statement and appending an assigned certainty. Certainty factors range from -5 (positive that a situation does NOT hold) to 5 (positive that it does hold). All assertions made by PROSPECTOR have associated certainty factors. The program now enters the consequent mode of reasoning.

Presently, the models in PROSPECTOR and their scores are:

1 Komatiitic nickel sulfide deposit -- KNSD (0.0)
2 Type-A porphyry copper deposit -- PCDA (0.0) **current hypothesis**
3 Kuroko-type massive sulfide deposit -- MSD (.001999)
4 Western-states-type sandstone uranium deposit -- WSSU (0.0)
5 Mississippi-Valley-type lead/zinc deposit -- MVTD (-4.8)

Which of the above would you like to rule out? NONE

The volunteered information was sufficient to rule out one of the models, but was not nearly specific enough to establish any of the others. The choice of the second model as the current hypothesis was based not on its certainty, but on the number and types of connections between it and the volunteered statements. To avoid a possibly fruitless investigation of unpromising models, the system gives the user the opportunity to exclude models of lesser interest. In this case, the user allows all of the models to be explored, and the system begins with the current hypothesis.

I am considering the possibility of a
(Type-A porphyry copper deposit -- PCDA)
in the target area.

The following questions are intended to establish the nature of the petrotectonic setting.

1--To what degree do you believe that:
(the target area is in a continental margin mobile belt) ? -2

This is a discouraging answer for the PCDA model, and it removes a context needed for the investigation of other properties of the continental margin mobile belt.

The following questions are intended to determine the nature of the

regional environment.

2--To what degree do you believe that:
(there are granitic intrusives in the region) ? 5

3--To what degree do you believe that:
(the region contains an abundance of small stocks) ? -3

.
.
.

8--To what degree do you believe that:
(igneous rocks in the region have porphyritic texture) ? WHY

I am trying to establish whether some of the intrusive rocks in your area have textures suggestive of a hypabyssal to subvolcanic environment. Porphyry textures resulting from the rapid cooling and crystallization of a silicate melt which contains large mineral grains are suggestive of relatively shallow levels of emplacement.

Like MYCIN, PROSPECTOR provides a variety of commands for doing such things as requesting a summary, changing previous answers, or volunteering additional formation. The above response to a WHY command is not automatically generated from the rule but is merely stored text associated with the space.

8--To what degree do you believe that:
(igneous rocks in the region have porphyritic texture) ? 5

My certainty in (the texture and morphology of the intrusive system suggest a subvolcanic (hypabyssal) regional environment) is now: 4.924

Do you want to pursue this further? NO

At this point the system noted that the certainty for the hypothesis about a hypabyssal environment was above 4.5, and it also noted that if the remaining rules were answered in the most unfavorable way possible, the certainty could not fall below 4.5. To save time, it let the user decide whether or not to continue working on this subgoal.

The questioning continues as above until it becomes necessary to consider properties that vary within the target area. Thus, for the first time it becomes necessary to be able to use the rules more than once, and a communication convention must be established to distinguish different spatial locations.

Porphyry copper deposits are often composed of characteristic zones of alteration and mineralization. (In some cases these can be looked upon as different stages superimposed upon one another.) I shall call the most prominent such zone Zone 1. Even though you may have already identified this zone as, say, the propylitic zone or the sericitic zone, I need to consider other possibilities as well. Thus, the following questions are

intended to determine the possibilities for Zone 1.

25--To what degree do you believe that:
(there are fresh-looking feldspars in Zone 1) ? -5

.
.
.

Questioning continues in this manner. Eventually, PROSPECTOR establishes that two zones exist in the region of interest, and after examining each in detail, it concludes that their geologic makeup supports the hypothesis that Type-A porphyry copper exists in the region.

On a scale from -5 to 5, my certainty that (there are alteration zones that are favorable for a Type-A porphyry copper deposit) is now 4.833

There are two favorable factors; in order of importance:

2.3.1: 1. there is a sericitic zone in Zone 1 (4.834)
** establishes 2.3.1 **

2.3.1: 2. there is a potassic zone in Zone 2 (4.499)

There are many factors that would have been unfavorable, had no favorable factor existed to override them; in order of importance:

2.3.1: 3. there is a barren-core zone in Zone 1 (-5.0)

2.3.1: 4. there is a potassic zone in Zone 1 (-5.0)

2.3.1: 5. there is a propylitic zone in Zone 1 (-5.0)

2.3.1: 6. there is a sericitic zone in Zone 2 (-5.0)

2.3.1: 7. there is a propylitic zone in Zone 2 (-4.989)

2.3.1: 8. there is a barren-core zone in Zone 2 (-4.495)

For which of the above do you wish to see additional information?

Knowledge representation

The network structure that is used to represent the geological knowledge embodied in PROSPECTOR is called the inference network, which guides the plausible reasoning performed by the system. The nodes in this network correspond to various assertions, such as "There is pervasively biotized hornblende" or "There is alteration favorable for the potassic zone of a porphyry copper deposit." In a particular run, any assertion may be known to be true, known to be false, or suspected to be true with some probability.

Most of the arcs in the inference network define *inference rules* that specify how the probability of one assertion affects the probability of another assertion. For example, the presence of pervasively biotized hornblende suggests the potassic zone of a porphyry copper deposit, and the absence of any biotized hornblende is very discouraging for that conclusion. These inference rules correspond to the production rules used in MYCIN. The remaining arcs indicate that an assertion is the "context" for another assertion, preventing conclusions from being drawn until the right contexts are established. For example, one should establish that hornblende has been altered to biotite before asking about the degree of alteration.

The primary task confronting a geologist who wants to prepare a new model for PROSPECTOR is the representation of his or her model as an inference network. The current system contains models of five different types of deposits, developed in cooperation with five different consulting geologists. The following statistics give a rough indication of the size and complexity of these models.

Model	Number of Assertions	Number of Rules
Koroko-type massive sulfide	39	34
Mississippi-Valley-type lead/zinc	28	20
Type-A porphyry copper	187	91
Komatiitic nickel sulfide	75	49
Roll-front sandstone uranium	212	133
Total:	541	327

To allow certain kinds of logical reasoning by the system, each assertion is represented as a "space" in a partitioned semantic network (Hendrix, 1975a). A typical space asserts the hypothetical existence of physical entities having specific properties (such as being composed of biotite) and participating in specific relations (such as an alteration relation). In addition, a large taxonomic network describes important element/subset relations among the terms mentioned, such as the fact that biotite is a mica, which in turn is a silicate, which in turn is a mineral.

The articulation of assertions as a set of relations allows the system to recognize subset/superset connections between pairs of assertions. For example, the assertion that "There is pervasively biotized hornblende" is clearly related to the assertion that "There is mica"; assertion of the first also asserts the second, and denial of the second denies the first. This kind of recognition is used in two main ways. First, it provides important intermodel and intramodel connections beyond those given explicitly by the inference rules. Second, it

allows the system to recognize connections between information volunteered by the user and the coded models.

Probabilistic reasoning

Some of the logical constraints that exist between spaces have probabilistic implications. In particular, if A is an instance of (subset of) B, then the probability of A can never exceed the probability of B. We maintain this constraint by automatically generating certain inference rules. For example, if evidence E could raise the probability of A above the probability of B, then we generate a rule from E to B that will increase the probability of B sufficiently to just satisfy the constraint. The exact procedure used here is described in Duda et al., 1977.

Since the various inference rules interconnect to form an inference network, when the user provides some evidence this information can change the probabilities of several hypotheses, which in turn can change the probabilities of hypotheses that depend upon them. The probability formulas determine exactly how these probability changes propagate through the inference net. (The reader might also refer to the handbook articles on IRIS and CASNET for other discussions of propagation.)

Control

As mentioned earlier, PROSPECTOR is a mixed-initiative system that begins by allowing the user to volunteer information about the prospect. This volunteered information is currently limited to simple statements in constrained English about the names, ages, and forms of the rocks and the types of minerals present. These statements are parsed by LIFER--a natural language interface facility developed by Hendrix (1977)--and represented as partitioned semantic networks. A network matching program compares each of these volunteered spaces against the spaces in the models, noting any subset, superset, or equality relations that occur.

If a volunteered space is exactly equal to a space in a model, the probability of the model space is updated and that change is propagated through the inference network. If a volunteered space is a subset of a space in a model and if it has a higher probability than the model space, then once again the probability of the model space is updated and that change is propagated through the inference network.

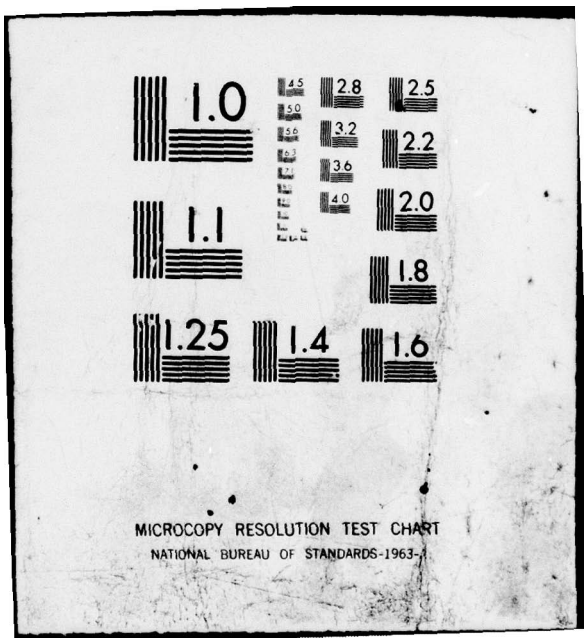
Unfortunately, if the volunteered space matches a superset of a model space (which usually occurs), no probability change can be made unless the user expresses doubt about the situation. For example, if the user mentions biotite, the probability of the space that asserts that there is pervasively biotized hornblende is unchanged, unless the user has said that he or she doubts that there is any biotite. However, it is obvious that the system may want to follow up this observation, and the existence of the connection to the model is recorded.

When the user has finished the initial volunteering, PROSPECTOR scores the various models on the basis of the number and types of connections that have occurred and selects the best matching model for further investigation. Here the basic control strategy is MYCIN-

AD-A076 875 STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE F/G 9/4
APPLICATIONS-ORIENTED AI RESEARCH: SCIENCE AND MATHEMATICS.(U)
AUG 79 J S BENNETT , B G BUCHANAN MDA903-77-C-0322
UNCLASSIFIED STAN-CS-79-756 NL

2 OF 2
AD
A076875

END
DATE
FILMED
12-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

like *backward chaining* or *consequent reasoning*. At any given time there is a current goal space whose existence is to be determined. The initial goal space is the one that corresponds to the best matching model. The various spaces in the models either represent evidence that can be sought from the user (are "askable") or internal hypotheses that are to be deduced from evidence (are "unaskable"). Naturally, the initial goal space is always unaskable. If the current goal space has any unestablished context spaces, they are pushed on the goal stack and one of them becomes the new current goal.

If the current goal is askable and has not been asked before, the user is asked about it; the effects of the answer are propagated through the inference network; and the process is repeated. If it is unaskable, it must be either the consequence of one or more inference rules or a logical combination of one or more other spaces. In the former case, the rules are scored to determine their potential effectiveness in influencing H, and the antecedent of the best scoring rule becomes the next goal. In the latter case a predetermined supporting space becomes the next goal. In either case the same procedure is repeated until either: (a) The top-level goal becomes so unlikely that another top-level goal is selected, (b) all of the askable spaces have been asked, or (c) the user interrupts with new volunteered information.

Summary

This brief overview covers the basic knowledge representation and inference mechanisms used in PROSPECTOR. Many aspects of the system have not been mentioned, such as the treatment of quantitative evidence, the matching procedure, the use of graphical input, the inference network compiler, the explanation system, model acquisition aids, and the test and evaluation effort.

The five models in the current system are but a fraction of what is needed for comprehensive coverage, and even these models have only recently achieved the degree of completeness required for doing meaningful evaluations. Limited initial tests have shown very close agreement between the evaluations provided by the system and the evaluations of the model designers, using data from actual deposits of the types modeled. More information on the system, the extent of its geological knowledge, its performance on known deposits, and its possible applications can be found in Duda et al., 1978.

References

See Duda, Hart, and Nilsson (1976), Duda et al. (1977), Duda et al. (1978), Hendrix (1975a), Hendrix (1977), Pople, Myers, and Miller (1975), and ZADEH (1965).

References

Anderson, R. H. The use of production systems in RITA to construct personal computer *agents*. Proceedings of the Workshop on Pattern-directed Inference Systems, SIGART Newsletter (No. 63), 1977, pp. 23-28.

Anderson, R. H., Gallegos, M., Gillogly, J. J., Greenberg, R., and Villanueva, R. RITA Reference Manual (R-1808-ARPA). Santa Monica, Ca.: The Rand Corp., September 1977.

Anderson, R. H., and Gillogly, J. J. Rand Intelligent Terminal Agent (RITA): Design philosophy (R-1809-ARPA). Santa Monica, Ca.: The Rand Corp., February 1976. (a)

Anderson, R. H., and Gillogly, J. J. The Rand Intelligent Terminal (RITA) as a network access aid. AFIP Proceedings, 1976, 45, 501-509. (b)

Banks, J. E. Naming Organic Compounds. Philadelphia: W. B. Saunders Co., 1976.

Bennett, J. S., Creary, L. A., Engelmore, R. M., and Melosh, R. E. SACON: A Knowledge-based consultant in Strucutral Analysis, Stanford Heuristic Programming Project, Report No. 78-23, 1978.

Bledsoe, W. W. Splitting and Reduction Heuristics in Automatic Theorem Proving. Artificial Intelligence, 1971, 2, p. 73.

Brotz, D. Embedding Heuristic Problem Solving Methods in a Mechanical Theorem Prover. Stanford University, Computer Science Dept., Rep. No. STAN-CS-74-443, August 1974.

Brown, H., and Masinter, L. An Algorithm for the Construction of the Graphs of Organic Molecules. Discrete Mathematics, 1974, 8, 227.

Brown, H., Masinter, L., and Hjelmeland, L. Constructive Graph Labeling Using Double Cosets. Discrete mathematics, 1974, 7, 1.

Brown, J. S., and Traub, J. F. On Euclid's Algorithm and the computation of polynomial greatest common divisors. JACM, 1971, 18(4), 505-514.

Buchanan, B. G. Applications of Artificial Intelligence to Scientific Reasoning, Second USA-Japan Computer Conference. AFIPS and IPSJ, Tokyo, 1975, pp. 189-194.

Buchanan, B. G., and Feigenbaum, E. A. DENDRAL and Meta-DENDRAL: Their applications dimension. Journal of Artificial Intelligence, 1978, 11(1,2), 5-24.

Buchanan, B. G., Sutherland, G. L., and Feigenbaum, E. A. Some problems of artificial intelligence in the context of organic chemistry. Machine Intelligence 5, Edinburgh University Press, 1969.

Buchanan, B. G. Scientific theory formation by computer. In J. C. Simon (Ed.),Proceedings of NATO Advanced Study Institute on Computer Oriented Learning Processes, Noordhoff, Leydon, 1976.

Buchanan, B. G., Duffield, A. M., Robertson, A. V., An Application of Artificial Intelligence to the Interpretation of Mass Spectra. In G. W. A. Milne (Ed.), Mass Spectrometry Techniques and Appliances. New York: John Wiley and Sons, 1971. P. 121.

Buchanan, B. G., Smith, D. H., White, W. C., Gritter, R. J., Feigenbaum, E. A., Lederberg, J., and Djerassi, C. Application of AI for Chemical Inference XXII. Automatic rule formation in mass spectrometry by means of the Meta-DENDRAL program. Journal of the American Chemical Society, 1976, 98(20), 6168-6178.

Buchanan, B. G., Sutherland, G. L., and Feigenbaum, E. A. Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry. In B. Meltzer and D. Michie (Eds.), Machine Intelligence 4. Edinburgh: University Press, 1969.

Buchs, A., Delfino, A. B., Duffield, A. M., Djerassi, C., Buchanan, B., Feigenbaum E. A., and Lederberg, J. Applications of Artificial Intelligence for Chemical Inference VI. Approach to a General Method of Interpreting Low Resolution Mass Spectra with a Computer. Helvetica Chimica Acta, 1970, 53, 1394.

Buchs, A., Duffield, A. M., Schroll, G., Djerassi, C., Delfino, A. B., Buchanan, B. G., Sutherland, G. L., Feigenbaum, E. A., and Lederberg, J. Applications of Artificial Intelligence For Chemical Inference IV. Saturated Amines Diagnosed by Their Low Resolution Mass Spectra and Nuclear Magnetic Resonance Spectra. Journal of the American Chemical Society, 1970, 92, 6831.

Carhart, R. E, and Smith, D. H. Applications of Artificial Intelligence for Chemical Inference XX. *Intelligent* Use of Constraints in Computer-Assisted Structure Elucidation. Computers and Chemistry, 1976, 1, 79.

Carhart, R. E., Smith, D. H., Brown, H., and Djerassi, C. Applications of Artificial Intelligence for Chemical Inference XVII. An Approach to Computer-Assisted Elucidation of Molecular Structure. Journal of the American Chemical Society, 1975, 97, 5755.

Cheer, C., Smith, D. H., Djerassi, C., Tursch, B., Braekman, J. C., and Daloze, D. Applications of Artificial Intelligence for Chemical Inference XXI. Chemical Studies of Marine Invertebrates XVII. The Computer-Assisted Identification of [+]-Palustrol in the Marine Organism Cespitularia sp., aff. Subvirdis. Tetrahedron, 1976, 32, 1807.

Churchman, C. W., and Buchanan, B. G. On the Design of Inductive Systems: Some Philosophical Problems. British Journal for the Philosophy of Science, 1969, 20, 311-323.

Codd, E. F. Seven Steps to Rendezvous with the Casual User. In J. W. Klimbie and K. I. Koffeman (Eds.), Data Base Management. New York: North Holland, 1974. Pp. 179-200.

Cory, E. J., and Wipke, W. T. Computer assisted design of complex organic synthesis. Science, 1969, 166, 178-192.

Davis, R. Applications of Meta-Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases. Stanford AI Lab, Memo AIM-283, 1976.

Davis, R. Interactive transfer of expertise: Acquisition of new inference rules. IJCAI 5, 1977, 321-328.

Davis, R., Knowledge acquisition in rule-based systems: Knowledge about representations as a basis for system construction and maintenance. In D. Waterman and F. Hayes-Roth (Eds.), Pattern-directed Inference Systems. New York: Academic Press, 1978. Pp. 99-134.

Davis, R., and Buchanan, B. Meta-level knowledge: Overview and Applications. IJCAI 5, 1977, 920-928.

Duda, R. O., Gaschnig, J., Hart, P. E., Konolige, K., Reboh, R., Barrett, P., and Slocum, J. Development of the PROSPECTOR consultation system for mineral exploration. Final Report, SRI Projects 5821 and 6415. SRI International, Inc., Menlo Park, Calif., 1978.

Duda, R. O., Hart, P. E., and Nilsson, N. J. Subjective Bayesian methods for rule-based inference systems. AFIPS, 1976, 45, 1075-1082.

Duda, R. O., Hart, P. E., Nilsson, N. J., Reboh, R., Slocum, J., and Sutherland, G. L. Development of a computer-based consultant for mineral exploration. Annual Report, SRI Projects 5821 and 6415. SRI International, Inc., Menlo Park, Calif., 1977.

Duffield, A. M., Robertson, A. V., Djerassi, C., Buchanan, B. G. Sutherland, G. L., Feigenbaum, E. A., and Lederberg, J. Application of Artificial Intelligence for Chemical Inference II. Interpretation of Low Resolution Mass Spectra of Ketones. Journal of the American Chemical Society, 91(11), 1969.

Engelmore, R. E., and Terry, A. Structure and function of the CRYSALIS system. IJCAI 6, Tokyo, 1978, 250-256.

Engelmore, R. S., and Nii, H. P. A Knowledge-based System for the Interpretation of Protein X-ray Crystallographic Data. Stanford Heuristic Programming Project Report HPP-77-2, Computer Science Dept., 1977.

Erman, L. D. Overview of the HEARSAY speech understanding research. Working Papers in Speech Recognition-IV--The HEARSAY II System, Carnegie-Mellon University, Computer Science Speech Group, 1976.

Evans, M. A Program for the Solution of Geometric-Analogy Intelligence Test Questions. In M. Minsky (Ed.), Semantic Information Processing. Cambridge: MIT Press, 1968. Pp. 271-353.

Fahlman, S. E. A System for representing and using real world knowledge. Doctoral dissertation, MIT AI Lab, September 1977.

Fateman, R. J. Essays in Algebraic Manipulation. TR-95, MIT Computer Science Lab, April 1972.

Fateman, R. J. An Approach to Automatic Asymptotic Expansions. Proc. of an ACM Symposium on Symbolic and Algebraic Computation, August 1976.

Feigenbaum, E. A. , Engelmore, R. S., and Johnson, C. K. A Correlation Between Crystallographic Computing and Artificial Intelligence Research. Acta Crystallographica, 1977, A33, 13.

Feigenbaum, E. A. The art of artificial intelligence: Themes and case studies in knowledge engineering. IJCAI 5, 1977, 1014-1029.

Feigenbaum, E. A. , and Buchanan, B. Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry. In B. J. Kinariwala and F. F. Kuo (Eds.), Proc. Hawaii Int. Conf. on System Sciences, University of Hawaii Press, 1968.

Feigenbaum, E. A. , Buchanan, B., and Lederberg, J. On Generality and Problem Solving: A Case Study Using the DENDRAL Program. In Meltzer and Michie (Eds.), Machine Intelligence 6. New York: American Elsevier, 1971. Pp. 165-190.

Gelernter, H. L. Realization of a Geometry-Theorem Proving Machine. In E. A. Feigenbaum and Feldman (Eds.), Computers and Thought. New York: McGraw-Hill, 1963. Pp. 134-152.

Gelernter, H. L., Sanders, A. F., Larsen, D. L., Agarival, K. K., Boivie, R. H., Spritzer, G. A., and Searleman, J. E. Empirical explorations of SYNCHEM. Science, 1977, 197(4308), 1041-1049.

Genesereth, M. R. DB: A high level data base system with inference, Memo 4, MACSYMA Group, MIT, December 1976.

Genesereth, M. R. The difficulties of using MACSYMA and the function of user aids. Proc. of the 1st MACSYMA Users' Conference, NASA Report CP-2012, July 1977.

Genesereth, M. R. Automated Consultation for Complex Computer Systems. Doctoral dissertation, Harvard University, September 1978.

Gosper, R. W. Indefinite hypergeometric sums in MACSYMA. Proc. of the 1st MACSYMA Users' Conference, NASA Report CP-2012, July 1977.

Grossman, R. Some Data Base Applications of Constraint Expressions. TR-168, MIT Computer Science Lab, February 1976.

Gund, P., Andose, J. D., and Rhodes, J. B Computer assisted analysis in drug research. In W. T. Wipke and W. J. Howe (Eds.), Computer-assisted Organic Synthesis. Washington, D. C.: American Chemical Society, 1977. Pp. 179-187.

Hadamard, J. The Psychology of Invention in the Mathematical Field. New York: Dover, 1945.

Harris, L. R. ROBOT: A high performance natural language processor for data base query. SIGART Newsletter (No. 61), Feb. 1977, pp. 39-40.

Hart, P. E. Progress on a Computer Based Consultant. IJCAI 4, 1975, 831-841.

Hayes-Roth, F., and Lesser, V. R. Focus of attention in a distributed-logic speech understanding system. Proc. of IEEE, Int. Conf. on ASSP, Philadelphia, Pa., 1976.

Hearn, A. REDUCE 2 User's Manual. Stanford AI Project Memo AI-90, May 1969.

Hearn, A. REDUCE 2: A system and language for algebraic manipulation. Proceedings of the 2nd Symposium on Symbolic and Algebraic Manipulation, March 1971.

Hearn, A. C. REDUCE 2 Users' Manual. Univ. of Utah Computational Physics Group, Report No. UCP-19, March 1973.

Heathcock, C. H., and Clark, R. D. Journal of Organic Chemistry, 1976, 41, 636-643.

Hedrick, C. A computer program to learn production systems using a semantic net. Doctoral dissertation, Graduate School of Industrial Admin., Carnegie-Mellon, July 1974.

Heiser, J. A computerized Psychopharmacology Advisor. SUMEX Annual Report, Computer Science Dept., Stanford University, 1977-1978.

Hempel, G. Fundamentals of Concept Formation in Empirical Science. Chicago: University of Chicago Press, 1952.

Hendrix, G. G. Expanding the utility of semantic networks through partitioning. IJCAI 4, 1975, 115-121. (a)

Hendrix, G. G. Human engineering for applied natural language processing. IJCAI 5, 1977, 183-191.

Hunt, E. B. Artificial Intelligence. New York: Academic Press, 1975.

Jurs, P. C. Chemical data interpretation using pattern recognition techniques. In W. T. Wipke, S. R. Heller, R. J. Feldman, and E. Hyde, (Eds.), Computer Representation and Manipulation of Chemical Information. New York: Wiley-Interscience, 1974. Pp. 265-285.

Kernighan, B. W., and Ritchie, D. M. The C Programming Language. New Jersey: Prentice Hall, 1978.

Kling, R. Reasoning by Analogy with Applications to Heuristic Problem Solving: A Case Study. Memo AIM-147, CS-216, Stanford University, August 1971.

Knuth, D. Surreal Numbers. Reading, Mass.: Addison-Wesley, 1974.

Koestler, A. The Act of Creation. New York: Dell, 1967.

Laing, R. D. Rules and Metarules. In R. D. Laing (Ed.), The Politics of the Family and Other Essays. New York: Vintage Books, 1971. Pp. 103-116.

Lakatos, I. Proofs and Refutations: The Logic of Mathematical Discovery. London: Cambridge Univ. Press, 1976.

Lederberg, J. Computation of Molecular Formulas for Mass Spectrometry. San Francisco: Holden-Day, 1964. (a)

Lederberg, J. DENDRAL-64: A System for Computer Construction, Enumeration and Notation of Organic Molecules as Tree Structures and Cyclic Graphs. Part I. Notational algorithm for tree structures. NASA CR.57029, 1964. (b)

Lederberg, J. DENDRAL-64: Part II. Topology of cyclic graphs. NASA CR.68898, 1965. (a)

Lederberg, J. Systematics of organic molecules, graph topology and Hamilton circuits. A general outline of the DENDRAL system. NASA CR-48899, 1965. (b)

Lederberg, J. Topological Mapping of Organic Molecules. Proc. Nat. Acad. Sci., 1965, 53(1), 134-139. (c)

Lederberg, J. DENDRAL-64: Part III. Complete chemical graphs; embedding rings in trees. NASA, 1969.

Lederberg, J., and Feigenbaum, E. A. Mechanization of Inductive Inference in Organic Chemistry. In B. Kleinmuntz (Ed.), Formal Representations for Human Judgment. New York: John Wiley, 1968.

Lenat, D. B. The ubiquity of discovery. Journal of Artificial Intelligence, 1977, 9(3), 257-286.

Lenat, D. B. AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search. SAIL AIM-286, AI Lab, Stanford University, July, 1976.

Lewis, V. E. User aids for MACSYMA. Proc. of the 1st MACSYMA Users' Conference, NASA Report CP-2012, July 1977.

Lindsay, R., Buchanan, B. G., Feigenbaum, E. A., and Lederberg, J. DENDRAL. New York: McGraw Hill, forthcoming.

Masinter, L., Sridharan, N. S., Carhart, R., and Smith, D. H. Application of Artificial Intelligence for Chemical Inference XII: Exhaustive Generation of Cyclic and Acyclic Isomers. Journal of the American Chemical Society, 1974, 96, 7702.

Mathlab Group MACSYMA Reference Manual. MIT Computer Science Lab, December 1977.

Minsky, M. Steps toward artificial intelligence. In E. A. Feigenbaum and J. Feldman (Eds.), Computers and Thought. New York: McGraw-Hill, 1963. Pp. 406-450.

Minsky, M. Frames. In P. Winston (Ed.), The Psychology of Computer Vision. New York: McGraw-Hill, 1975.

Mitchell, T. M. Version spaces: An approach to rule revision during rule induction. IJCAI 5, 1977, 305-310. (Also Heuristic Programming Project Memo HPP-77-13, Computer Science Dept., Stanford University, 1977.)

Mitchell, T. M., and Schwenzer, G. M. Applications of AI for chemical inference, XXV: A computer program for automated empirical 13C NMR rule formation. Organic Magnetic Resonance, 1978, 11(8), 378.

Morrill, K., Smith, D. H., and Djerassi, C. Computer-assisted Analysis of the High Resolution Mass Spectra of Macrolide Antibiotics. Organic Mass Spectrometry, 1977.

Moses, J. Symbolic Integration: The Stormy Decade. Communicatins of the ACM, 1971, 14(8), 548-560.

Moses, J. A MACSYMA Primer. Mathlab Memo No. 2, MIT Computer Science Lab, October 1975.

Moses, J., and Yun, D. Y. The EZGCD Algorithm. Proc. of the ACM National Convention, August 1973.

Musser, D. R. Multivariate polynomial factoring. JACM, 1975, 22(2), 291-307.

Mylopoulos, J. and Roussopoulos, N. Using Semantic Networks for Data Base Management. Proc. International Conf. on Very Large Data Bases, Framingham, Mass., Sept. 1975, pp. 144-172.

Nii, H. P., and Aiello, N. AGE (Attempt to Generalize): Profile of the AGE-0 System, Heuristic Programming Project, Working Paper HPP-78-5, Stanford University, June 1978.

Nilsson, N. (Ed.) Artificial Intelligence--Research and Applications. Stanford Research Institute, Inc., Menlo Park, Calif., 1975.

Norman, A. C. On Computing with Formal Power Series. Transactions on Mathematical Software (ACM), 1975, 1(4), 346-356.

Papert, S. Teaching Children to be Mathematicians Versus Teaching About Mathematics. Intl. Jour. Math Ed. Sci. Tech. 3, July-Sept. 1972, No. 3, pp. 249-262.

Pivar, M., and Finkelstein, M. Automation, using LISP, of Inductive Inference on Sequences. In E. C. Berkeley and D. G. Bobrow (Eds.), The Programming Language LISP: Its Operation and Applications. Cambridge: Information International, 1964.

Poincare, H. The Foundations of Science: Science and Hypothesis, The Value of Science, Science and Method. New York: The Science Press, 1929.

Polya, G. Mathematics and Plausible Reasoning (2 vols.). New York: John Wiley and Sons, 1954.

Popl, H. E., Jr., Myers, J. D., and Miller, R. A. DIALOG: A model of diagnostic logic for internal medicine. IJCAI 4, 1975, 848-855.

Risch, R. The Problem of Integration in Finite Terms. Trans. of the AMS, May 1969, 139.

Ritchie, D. M., and Thompson, K. The UNIX time-sharing system. Communications of the ACM, 1974, 17, 365-375.

Rothstein, M. A New Algorithm for the Integration of Exponential and Logarithmic Functions. Proc. of the 1st MACSYMA Users' Conference, NASA Report CP-2012, July 1977.

Sacerdoti, E. D. Language Access to Distributed Data With Error Recovery, AI Technical Note 140, SRI International, Inc., Menlo Park, Calif., February 1977.

Sagalowicz, D. IDA: An intelligent data access program, AI Tech. Note 145, SRI International, Inc., Menlo Park, Calif., June 1977.

Samuel, A. L. Some studies of machine learning using the game of checkers. In E. A. Feigenbaum and J. Feldman (Eds.), Computers and Thought. New York: McGraw-Hill, 1963. Pp. 71-105.

Schroll, G., Duffield, A. M., Djerassi, C., Buchanan, B. G. Sutherland, G. L., Feigenbaum, E. A., and Lederberg, J. Application of Artificial Intelligence for Chemical Inference III. Aliphatic Ethers Diagnosed by Their Low Resolution Mass Spectra and NMR Data. Journal of the American Chemical Society, 1969, 91, 7440.

Sheikh, Y. M., Buchs, A., Delfino, A. B., Schroll, G. Duffield, A. M., Djerassi, C., Buchanan, B., Sutherland, G. L., Feigenbaum, E. A., and Lederberg, J. Applications of Artificial Intelligence for Chemical Inference V. An Approach to the Computer Generation of Cyclic Structures. Differentiation Between All the Possible Isomeric Ketones of Composition, C6H10O. Organic Mass Spectrometry, 1970, 4, 493.

Simon, H. Does Scientific Discovery Have a Logic? Philosophy of Science, 1973, 40(4), 471-480.

Simon, H., and Kotovsky, K. Human Acquisition of Concepts for Sequential Patterns. Psychology Review, 1963, 70:6, 534-546.

Simon, H., and Lea, G. Problem solving and rule induction: A unified view (rev.). Complex Information Processing Working Paper 227, Carnegie-Mellon, June 1973.

Smith, D. H. Applications of Artificial Intelligence for Chemical Inference. XV. Constructive Graph Labelling Applied to Chemical Problems. Chlorinated Hydrocarbons. Analytical Chemistry, 1975, 47, 1176.

Smith, D. H., Buchanan, B. G., Engelmore, R. S., Duffield, A. M., Yeo, A., Feigenbaum, E. A., Lederberg, J., and Djerassi, C. Applications of Artificial Intelligence for Chemical Inference VIII. An Approach to the Computer Interpretation of the High Resolution Mass Spectra of Complex Molecules. Structure Elucidation of Estrogenic Steroids. Journal of the American Chemical Society, 1972, 94, 5962.

Smith, D. H., Buchanan, B. G., Engelmore, R. S., Adlercreutz, H. and Djerassi, C. Applications of Artificial Intelligence for Chemical Inference IX. Analysis of Mixtures Without Prior Separation as Illustrated for Estrogens. Journal of the American Chemical Society 1973, 95, 6078.

Smith, D. H., and Carhart, R. E. Applications of Artificial Intelligence for Chemical Inference XXIV. Structural Isomerism of Mono- and Sesquiterpenoid Skeleton 1,2-. Tetrahedron, 1976, 32, 2513.

Smith, D. H., and Carhart, R. E. Structure Elucidation Based on Computer Analysis of High and Low Resolution Mass Spectral Data. In M. L. Gross (Ed.), High Performance Mass Spectrometry: Chemical Applications. Wash., D. C.: American Chemical Society, 1978. P. 325.

Smith, D. H., Konopelski, J. P., and Djerassi, C. Applications of Artificial Intelligence for Chemical Inference. XIX. Computer Generation of Ion Structures. Organic Mass Spectrometry, 1976, 11, 86.

Smith, R. G., Mitchell, T. M., Chestek, R. A., Buchanan, B. G. A model for learning systems. IJCAI 5, 1977, 338-343. (Also Heuristic Programming Project Memo HPP-77-14, Computer Science Dept., Stanford University, 1977.)

Sowa, J. F. Conceptual Graphs for a Data Base Interface. IBM Journal of Research and Development, 1976, 20(4), 336-357.

Sridharan, N. S. AI Journal: Special issue on applications in the sciences and medicine, 1978, 11(1,2), 1-195.

Streitwieser, A., and Heathcock, C. H. Introduction to Organic Chemistry. New York: MacMillan, 1976.

Thompson, F. B., and Thompson, B. H. Practical Natural Language Processing: the REL System as Prototype. In M. Rubinoff and M. C. Yovits (Eds.), Advances in Computers, 13. New York: Academic Press, 1975. Pp. 109-168.

Trager, B. M. Integration of Algebraic Functions. Doctoral dissertation, MIT Computer Science Lab, 1978.

Varkony, T. H., Carhart, R. E., and Smith, D. H. Computer Assisted Structure Elucidation, Ranking of Candidate Structures, Based on Comparison Between Predicted and Observed Mass Spectra. Paper presented at the ASMS meeting, Washington, D.C., 1977.

Varkony, T. H., Smith, D. H., and Djerassi, C. Computer-assisted structure manipulation: Studies in the biosynthesis of natural products. Tetrahedron, 1978, 34, 841-852.

Walker, D. E., Erman, L. D., Newell, A., Nilsson, N. J., Paxton, W. H., Winograd, T., and Woods, W. A. An Overview of Speech Understanding Research at SRI. IJCAI 5, 1977, 970-974.

Waltz, D. L. Natural language access to a large database: An engineering approach. IJCAI 4, 1975, 868-872.

Wang, H. Toward Mechanical Mathematics. IBM Jour. Research and Development 4, 1960, 1, 2-22.

Wang, P., and Rothschild, L. Factoring Multivariate Polynomials over the Integers. Math. of Comp., July 1975, 29, 935-950.

Waterman, D. A. Generalization learning techniques for automating the learning of heuristics. Artificial Intelligence, 1, 1970, 121-170.

Waterman, D. A. Adaptive production systems. Complex Information Processing Working Paper 285, Dept. of Psychology, Carnegie-Mellon, December 1974.

Waterman, D. A. Exemplary programming in RITA. In D. Waterman and F. Hayes-Roth (Eds.), Pattern-directed Inference Systems. New York: Academic Press, 1977. Pp. 261-279. (a)

Waterman, D. A. Rule-directed Interactive Transaction Agents: An Approach to Knowledge Acquisition (R-2171-ARPA). Santa Monica, Ca.: The Rand Corp., 1977. (b)

Waterman, D. A., and Jenkins, B. Heuristic Modeling Using Rule-based Computer Systems (P-5811). Santa Monica, Ca.: The Rand Corp., 1977.

Winston, P. H. Artificial Intelligence. Reading, Mass: Addison-Wesley, 1977.

Winston, P. H. Learning structural descriptions from examples. Doctoral dissertation (AI-TR-231), MIT, September 1970.

Wipke, W. T., Braun, H., Smith, G., Choplin, F., and Sieber, W. SECS--Simulation and evaluation of chemical synthesis: Strategy and planning. In W. T. Wipke and W. J. House (Eds.), Computer-assisted Organic Synthesis. Washington D. C.: American Chemical Society, 1977. Pp. 97-127.

Woods, W. A. Progress in natural language understanding, An application to lunar geology. Proc. NCC, 42, AFIPS, 1973, 441-450.

Zadeh, L. A. Fuzzy sets. Information and Control, 1965, 8, 338-353.

Zippel, R. Univariate Power Series Expansions in Algebraic Manipulation. Proc. of an ACM Symposium on Symbolic and Algebraic Computation, August 1976.

Index